Universidad Nacional Autónoma de México

Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Facultad de Ingeniería
Laboratorio de Biorobótica

Localización y seguimiento de objetos a través de sus características principales

Ing. Rommel Sánchez Verdejo
rommel@cuevano.org
2008

a

Resumen

El objetivo de este trabajo es proporcionar al Laboratorio de Biorobótica del Posgrado de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, un sistema de visión que sea fácil de modificar para futuras generaciones, de un conjunto de herramientas en mantenimiento constante, así como perfectamente documentadas, utilizando implementaciones con su respectivo código, anteriormente propuesto, evaluando y mejorando su rendimiento a través de técnicas de optimización por hardware y una técnica teórica, en este caso: la cuantización vectorial.

En este trabajo se presenta una comparación entre dos técnicas para la correspondencia de características locales mencionadas en la bibliografía: correspondencia por distancia Euclideana y correspondencia por cuantización vectorial, utilizando el algoritmo SURF [BTG06] como generador de características locales llamadas descriptor.

Agradacimientos

Papás: quién me han soportado durante 27 años sin exigir nada a cambio más allá de mi bienestar constante.

Amigos: los que me conocen sabrán a qué me refiero: gracias por estar aquí.

Universidad: Una de las pocas cosas por las que me puedo sentir orgulloso de ser mexicano: gracias por todos estos años de preparacion. Gracias a todas las personas, maestros y compañeros, que hicieron de buena o mala manera, la persona que soy. Donde quiera que esté, donde quiera que pueda recordar:

`Por mi raza hablará el espíritu`

CONACyT: por sostenerme económicamente por 1.5 años.

Software Libre: Todas las personas que contribuyen de una u otra forma a él, ya que de no existir, no podría encajar en este mundo.

Paul's Family: Jerry & Nancy : You know, without your support, I would have never reached this far.

A mis sinodales: Jesús Savage Carmona, Francisco García Ugalde, Edgar Garduño Ángeles, María Elena Martínez Pérez y Boris Escalante Ramírez quienes pacientemente mejoraron la calidad de estre trabajo y con eso mi persona.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

El campo conocido ahora como visión por computadora o visión de máquina, es una de las áreas de mayor interés en el ámbito de la ciencia e ingeniería de la computación. Esta rama se consideraba una simple automatizacón o extensión de los temas estudiados por el Procesamiento digital de imágenes (PDI) . Sin embargo, el conjugar estos tópicos con otros como lo son, la Inteligencia Artificial (IA), el Reconcimiento de Patrones (RP) o la robótica, permiten explorar todo un rango de posibilidades, muchas de ellas siendo borde de conocimiento, aún por explorar, adquierie un sentido de mayor complejidad y formalismo en su estructura y divulgación.

El trabajo que se ha desarrollado con mayor intensidad durante los últimos años está enfocado en automatizar de manera eficiente alguna de las aplicaciones más comunes en el campo de la visión por computadora: localización y seguimiento de objetos, reconstrucción 3D de un objeto, determinación espacial de la cámara según su entorno[1], reconocimiento de rostros o bien recuperación de la infomación inscrita en un objeto.

En un sistema de visión por computadora (en adelante sólo me referiré a él como sistema de visión) la entrada es un conjunto de imágenes digitales, es decir una representación digital del entorno a través de un transductor, que en este caso es un sensor digital conocido como Coupled Charged Device (CCD)[2]. Estos sensores digitales se encuentran en cualquier cámara digital; en todo el proceso de adquisición se generan distorsiones, de las que se hablará con mayor detalle en los capítulos subsecuentes. Estas distoriones son generalmente transformaciones de carácter geométrico que pueden ser modeladas de cierta manera permitiendo así generar una reconsotrucción virtual bastante aproximada a la realidad del objeto, ya sea en dos dimensiones o bien en tres dimensiones.

[1] *SLAM*, por sus siglas en inglés
[2] sic

El objetivo de este trabajo esta en realizar un sistema que permita la localización automática de objetos previamente reconocidos a través de sus así llamadas características principales, a pesar de las deformaciones encontradas en el proceso de adquisición. El detalle de cómo se obtienen estas características será también materia de este estudio, ya que se implementaron algunas herramientas encontradas en la literatura. Cabe mencionar que existe más de una manera en la que se puede caracterizar a un conjunto de valores como un vector de características principales. Básicamente se otorga este nombre debido a que las características que se extraen de un punto, pixel, son llamadas invariantes a las deformaciones mencionadas; es decir, a pesar de las transformaciones afines o proyectivas así como cambios en la iluminación, éstos valores no tienen un gran intervalo de variación.

El algoritmo que ha alcanzado mayor satisfacción en la bibliografía es: Scale Invariant Feature Transform (SIFT) [Low04] que se revisará, de manera general. Recientemente otro algoritmo usado para la descripción de características locales, que ha captado mayor atención es: Speed-Up Robustness Features (SURF) [BTG06], el cual ha sido implementado y utlizado en este trabajo.

Estos vectores característicos son una respuesta a un conjunto de estímulos sobre una región circundante a un pixel en una imagen. Algunos de estos estímulos buscan resaltar ciertas características, que para los autores de cada algoritmo resultan ventajosas para el estudio de sus propiedades, ya sea la formalización o modelado de una tranformada espacio-escala, o bien la respuesta al conjunto de intensidades acumuladas.

He mencionado que los vectores característicos son la respuesta a una región circundante a un pixel determinado, motivo por el que también la manera de localizar estos pixeles resulta de suma importancia. De una u otra forma, se busca utilizar las características intrínsecas de cada pixel para determinar la localización de alguno de ellos que pudiera ser etiquetado como importante, asumiendo que estas características pudieran estar presentes cualquier otra imagen que contenga al objeto. Por lo que para poder obtener la localización de estos puntos un preprocesamiento se hace necesario. En este caso: *un detector de puntos importantes*. En la bibliografía revisada, el más usado es el detector de puntos de Harris/Stephens [HS88], derivado del detector de Moravec [Mor80], el cual ecuentra patrones con característica de escuadra, o lo que en una escena puede representar *esquinas*. La razón por la que es bastante usado y es considerado un buen detector es que computacionalmente resulta barato y siempre arroja buenos resultados de acuerdo a lo que veremos mas adelante. Cabe mencionar que en este trabajo se hace uso de otro detector el cual puede o no hacer

uso de la teoría descrita para el detector de puntos de Harris/Stephens; es denominado: *buenas características a seguir* [ST94]. Éste último es utilizado en este trabajo.

Una vez que ciertos puntos son localizados en una imagen, y éstos a su vez son caracterizados, faltaría determinar si alguno de estos puntos, con sus correspondientes vectores, pertenecen o no a determinada clase (objeto previamete entrenado). Para este proceso está el esquema habitual conocido simplemente como *apareamiento*, donde se busca si un vector está relativamente cerca de otro dada cierta métrica y bajo un criterio de búsqueda; En estre trabajo el criterio usado es la mínima distancia euclideana entre dos vectores siendo que esten el rango del más cercano y 0.7 veces el segundo más cercano: éste es el mejor acercamiento, como veremos en los resultados, pero como siempre, cualquier herramienta que ofrece la mejor solución también es costosa. Por ello, en esta tesis, se propone el uso de la cuantización vectorial como técnica de búsqueda reducida. Otros efoques, como lo son los árboles aleatrorios [DCM98], son comentados.

Por último, una vez que se ha encontrado y determinado el objeto, el seguimiento del mismo estará dado por un estimador de movimiento el cual utiliza un filtro de Kalman estándar [Gre06]. El trabajo realizado [Car07] en el laboratorio de BioRobótica[Bio], hace uso del filtro de Kalman del tipo *Unscented Kalman Filter (UKF)* por sus siglas en inglés. En este caso sólo se utilzará la versión estandarizada ya que se asume movimiento suave en el movimiento del objeto, así como también la implementación es sencilla y disponible en una de las bibliotecas que es usada en este trabajo.

El objetivo de este sistema de visión se centra en ser utilizado en una aplicación de robótica: tener un sistema de visión que permita a un robot, localizar objetos para el conjunto de pruebas determinadas por el Comité de la Liga @home en la competencia Robocup[Lea08]. Por ello, se hará mención de esta competencia y se mostrarán algunos resultados tangibles de la prueba.

Capítulo 2

Estado del arte

Realizando una restrospectiva en la bibliografía con mayor importancia por el número de citas en el campo de visión por computadora, específicamente en la descripción para características locales invariantes a escala [TM08] [Lin91], se ha trabajado en la búsqueda de modelos que van desde la curvatura de las formas, la forma en la que el sistema de visión humana trabaja, la detección de zonas del mismo color o la misma intensidad, pudiendo localizar líneas o circunferencias [DH72], detección de regiones como histogramas y dependencia de colores. El uso intenso de distintas técnicas del PDI, tal como la umbralización [SH85], filtrado, correspondencia por plantillas o el análisis binario, han logrado un gran avance. Desafortunadamente, ninguna de éstas o en su conjunto, había logrado resultados prácticos que pudieran ofrecer un rango acptable en el RP sobre todo cuando se presentan algunas deformaciones durante el proceso de adquisición y procesamiento. Se ha intentado utilizar herramientas como redes neuronales [MF94] en el proceso de decisión como una alternativa a esta necesidad. A pesar de esto, los resultados siguen dependiendo en gran medida de la intervención o decisión humana.[1]

Para responder a esta necesidad, se generó desde hace poco más de 10 años la teoría correspondiente no sólo a la detección de características globales, sino a una descripción robusta de las mismas, así como la descripción de características aún más finas [MS01]. Según Mikolajczyk y Tuytelaars [MS01], una región — en particular un punto — interesante tiene que cumplir con las siguientes características.

Repetitivo : Para una misma escena y dos tomas, en ambas, un gran porcentaje de regiones detectadas tienen que ser las mismas.

[1] Es un hecho, que aún con el sistema con menor índice o porcentaje de error — incluso siendo nulo — si de éste depende una vida humana: otro humano tiene que verificar el resultado. Siendo una discusión propia en ética profesional en la que se apela a la experiencia y juicio humano como la mejor decisión.

- *Robusto*: A pesar de las deformaciones, siempre se tienen que encontrar las mismas características

Discernible : Cada atributo del descriptor debe poder hacer clara la forma de encontrar su correspondencia.

Locales : Todos los atributos deberán ser extraidos por elementos locales (p. ej. pixeles vecinos).

Cantidad : Debe encontarse una gran cantidad de características

Precisión : Cada característica, debe poder ser localizable de manera única.

Eficiencia : Las aplicaciones siempre buscarán ejecutarse en tiempo real.

Según esta descripción, cualquier descriptor propuesto que cumpla con estas propiedades es conocido como un detector invariante de propiedades locales. Dado que la respuesta descrita en el vector de características puede ser en gran medida deformada [MS05] por transformaciones espaciales o de captura como lo es el cambio de escala, la rotación o la iluminación espacial, se ha trabajado en generar descriptores llamados *robustos*, incluso a las deformaciones con mayor grado de complejidad como pueden ser las deformaciones físicas del objeto.

2.1. Detectores de puntos

Los detectores de regiones son una forma particular de determinar una característica local que puede dar significado al contexto visual ya sea interpretado por el sistema de visión humana o de máquina. Un ejemplo puede ser la detección de bordes [HSSB98], o algún proceso de segmentación. Depende directamente de la aplicación el hacer que una región pueda ser expresada por un punto. Se puede mencionar el uso de regiones invariantes [MS02] a deformaciones afines, como un ejemplo de la detección de regiones y [MS01] como una evolución particular del mismo, por puntos.

En este trabajo, se toma el caso particular de un detector de puntos; por lo menos hasta este momento, no es importante considerar la deformación generada por la región circundante a cada punto sino sólo la posición y escala en la que se encuentra dentro de una escena dada. En algún momento, como desarrollo futuro de este sistema, se tendrá que considerar estas deformaciones utilizando regiones y no sólo puntos.

2.1.1. Harris y Stephens

El detector de Harris/Stephens [HS88] fue propuesto en 1988 como una modificación al detector de puntos y bordes de Moravec [Mor80], uno de los primeros detectores; sin embargo, hasta nuestros días es el más usado.

Ésta basado en la intensidad de los puntos y en una función de autocorrelación — solución propuesta a los problemas presentados por Moravec [Mor80] — que actúa sobre la región en la que se está utilizando el detector.
sea la función:

$$f(x,y) = \sum_{(x,y)} W(x,y)(I(x,y) - I(x_k + \Delta x, y_k + \Delta y))^2 \tag{2.1}$$

donde I es la imagen evaluada en la región W, $I(x,y)$ es el pixel localizado en la coordenada espacial (x,y) de la imagen I, $I(x_k, y_k)$ es una segunda imagen sobrepuesta a la primera en la posición $I(x,y)$, Δx y Δy son desplazamientos en la segunda imagen para determinar la correlacion entre ellas únicamente en la región W. Existen tres posibles valores que son interpretados de la siguiente manera :

- *Región Plana*: Intensidad constante. Los cambios deberían ser nulos.
- *Borde*: Cambio de signo. Sólo son detectados y considerados los cambios en dirección del borde (p. ej. Si el desplazamiento esta en una sola dirección o la forma en la que W sea aplicada).
- *Esquina*: Cambios altos, sin importar el desplazamiento o la dirección en la que se aplica W.

Un par de problemas que saltan a la vista son los altos cambios en presencia de ruido y la depencia en la dirección que se aplica W sobre la imagen. La solución propuesta consiste en extender el concepto de derivada. Desplazando una segunda imagen en Δx y Δy para después realizar la sustraccion de éstas, así, la función mencionada anteriormente se puede expresar de manera matricial:

$$\mathbf{A} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix} \tag{2.2}$$

donde cada elemento de la matriz se define como la segunda derivada de la imagen I localizada espacialmente en (x,y). Estas derivadas son aproximaciones sucesivas tomadas de un operador Sobel. Las derivadas se muestra en la ecuaciones 2.3 y 2.4.

$$D_x = I_x(x_k, y_k) * ((1,2,1)^T \otimes (1,0,-1)) \tag{2.3}$$

$$D_y = I_y(x_k, y_k) * ((1,0,-1)^T \otimes (1,2,1)) \tag{2.4}$$

siendo $*$ operador de convolución y $\otimes$ producto directo, Así :

$$D_{xx} = {D_x}^2 \tag{2.5}$$

$$D_{xy} = D_{xy} \tag{2.6}$$

$$D_{yy} = {D_y}^2. \tag{2.7}$$

La importancia que se asigna a cada uno de los vecinos pixeles vecinos en los que W esta ubicada, está determinada por una función de característica gaussiana W_g:

$$W_g(u,v) = \frac{1}{2\pi\sigma^2} e^{\frac{-(u^2+v^2)}{2\sigma^2}}. \tag{2.8}$$

la cual esta centrada en el pixel (x,y). La respuesta que se considera para la detección de puntos está determinado como :

$$R(x,y) = \texttt{Det}(A) - k\texttt{Tr}(A)^2 \tag{2.9}$$

donde k es una constante empírica determinada por [ST94] que cumple $0 < k < 0.25$, ya que experimentalmente [BL02] usar valores pequeños de k como valor fuerte para la detección de puntos.

2.1.1.1. Kanade y Tomasi

Este detector de puntos, también llamado Shi-Tomasi [ST94] o *"Good Features To Track"* [Cor01], está basado en el detector de Harris/Stephens [HS88]; utilizando la matriz de segundos momentos mostrada en la ecuación 2.2 pero la respuesta para la detección de puntos interesantes, estará en función de sus valores característicos[2]. De esta forma, $\forall(x,y) \in I(x,y)W(x,y)$, la respuesta que determina si el patrón corresponde o no a una interpretración de característica *esquina* esta determinado por:

$$R_e(x,y) = \texttt{min}(\lambda_1, \lambda_2) \tag{2.10}$$

Donde λ_1 y λ_2 son los valores característicos de la matriz de segundos momentos mostrada en 2.2 , $I(x,y)W(x,y)$ es la región W sobre I cuyo centro esta localizado en (x,y). Los puntos de interés están determinados por :

$$R(x,y) = \texttt{max}(R_e(x,y)) \tag{2.11}$$

es decir, el máximo valor en el conjunto definido por el mínimo de los valores característicos en la región W sobre I en el punto (x,y).

[2]Lo que es llamado *eigen-analisis*

2.1.2. Detección de máximos y mínimos en la representación espacio-escala

El algoritmo SIFT [Low04] encuentra puntos de interés a través de una aproximación a la Laplaciano de Gaussianas (LoG) por medio de Diferencia de Gaussianas (DoG), resultado generado por implementar el concepto de espacio-escala a través de una pirámide Gaussiana, siendo la diferencia entre cada nivel de la pirámide, una octava. El por qué utilzar DoG es un paso creativo que se describe a continuación:

$$\begin{aligned} D(x,y,\sigma) &= (L(x,y,k\sigma) - L(x,y,\sigma)) * I(x,y) && (2.12)\\ &\approx (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) && (2.13)\end{aligned}$$

donde $D(x,y,\sigma)$ es la representación espacio-escala de la imagen. $G(x,y,k\sigma)$ es la respuesta de la imagen original con un filtro gaussiano cuyo parámetro es σ y $L(x,y,k\sigma)$ es el Laplaciando de la imagen origial. Siendo que Lindeberg [MS01] demostró que el Laplaciano con el factor σ^2 es requerido para una verdadera invarianza en la representación espacio-escala, se genera la siguiente relación:

$$\begin{aligned} \frac{\partial G}{\partial \sigma} &\approx \sigma\nabla^2 G && (2.14)\\ \sigma\nabla^2 G &\approx \frac{G(x,y,k\sigma) - G(x,y,\sigma)}{k\sigma - \sigma} && (2.15)\\ G(x,y,k\sigma) - G(x,y,\sigma) &\approx (k-1)\sigma^2\nabla^2 G && (2.16)\end{aligned}$$

donde el valor de k es determinado empíricamente con un valor constante como $k = \sqrt{2}$. Cabe mencionar que el tipo de estructuras encontrado en este proceso 2.14, son las grandes estructuras binarias, es decir vecindarios donde existen el mismo tipo de características espaciales como lo es un rango común de intensidades.

La representacion espacio-escala se logra combinando dos tipo de representaciones comunes: una piramide gaussiana y la iteración consecuente de un filtro gaussiano. Cada nivel de la piramide gaussiana se denomina octava. En cada octava se convoluciona la imagen con un filtro gaussiano al que se hace variar distintos factores del parámetro σ en pasos de $\sigma + 3$ realizando entre cada par de valores asignados la diferencia correspondiente. Éste parámetro a partir de este momento se denotará como σ_L, puesto que corresponde al parámetro en el que se lleva a cabo la aproximación por DoG. El parámetro que se utiliza en cada peldaño de la pirámide gaussiana se hará referencia como σ unicamente.

Para llevar a cabo la localización de los puntos interesantes, se extraen el mínimo y el máximo de las DoG en una región de $3 \times 3 \times 3$: la cual incluye al pixel central y

a sus ocho vecinos en la escala correspondiente como también en las escalas inferior y superior. Es importante señalar que este procedimiento no es realizado en todos los máximos encontrados, ya que si éstos están lo suficientemente cercanos, serían inestables ya que las condiciones locales de un punto encontrado para cierta región es suficiente como para incluir otro que puede ser afectado facilmente por algunas deformaciones que se pudieran encontrar. El factor propuesto por Lowe [Low04] para realizar la búsqueda entre las distintas respuestas en cada octava es llamado frecuencia de muestreo y está determinado por $\sigma = 1.6$ en cada diferencia. Este valor de σ puede comenzar con $\sigma = 1.0$ a $\sigma > 5.8$ pero se deja a criterio de quien implemente el algoritmo.

Para obtener una mejor localización del punto en la región y eliminar las respuestas generadas por ruido, así como las respuestas por bajo contraste, tanto la posición como la escala se ajustan a una expresión cuadrática la cual es tomada con base en la expansión de Taylor de la función espacio-escala $D(x, y, \sigma)$ mostrada en la ecuación 2.12:

$$D(\hat{x}) = D + \frac{\partial^T}{\partial \hat{x}}\hat{x} + \frac{1}{2}\hat{x}^T \frac{\partial^2 D}{\partial \hat{x}^2}\hat{x}. \tag{2.17}$$

donde $\hat{x}$ es el vector (x, y, σ).

2.1.2.1. Harris tipo laplaciano

Una de las aportaciones de Lowe [Low03], es el haber generalizado el detector de puntos de Harris/Stephens en términos espacio-escala:

$$\mathcal{H}D(x, y, \sigma) \approx \sigma\nabla^2 G(x, y, \sigma) \approx \begin{bmatrix} \frac{\partial}{\partial x^2}D(x, y, \sigma) & \frac{\partial}{\partial x \partial y}D(x, y, \sigma) \\ \frac{\partial}{\partial x \partial y}D(x, y, \sigma) & \frac{\partial}{\partial y^2}D(x, y, \sigma) \end{bmatrix} \tag{2.18}$$

donde la $D(x, y, \sigma)$ es la aproximación de la LoG en función de la DoG mostrada en la ecuación 2.12, $\mathcal{H}$ es el operador matriz de segundos momentos o Hessiana.

El uso de la matriz de segundos momentos habiá sido mencionada anteriormente en la sección 2.1.1, sin embargo, en el detector de puntos de Harris/Stephens, no se hace referencia explícita a este operador. El umbral ultilizado por Harris/Stephens ahora es extendido para el uso en espacio-escala:

$$R(x, y) = \texttt{Det}(\mathcal{H}) - k\texttt{Tr}(\mathcal{H})^2.$$

Para eliminar la respuesta al ruido o a los bordes, se considera la expresión 2.2, tomando α como el máximo valor propio[3] y β como el mínimo valor propio[4], recordando la expresión 2.11, se define a r como la relación entre β y α, $\alpha = r\beta$ así:

[3] otra forma de llamarlo es *eigen-valor*.
[4] *Ibid.*

$$\frac{\mathtt{Tr}(\mathcal{H})^2}{\mathtt{Det}(\mathcal{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta\beta} = \frac{(r+1)^2}{r}$$

expresión que sólo está en función con la relacion de diferencia; dado la definición propuesta en la sección 2.1.1, si los cambios son iguales y altos, entonces se trata de una esquina, por lo que aquí se propone usar el siguiente umbral de decisión:

$$\frac{\mathtt{Tr}(\mathcal{H})^2}{\mathtt{Det}(\mathcal{H})} < \frac{(r+1)^2}{r} \tag{2.19}$$

.

2.1.3. Cálculo rápido de la hessiana

La propuesta de SURF [BTG06] está en detectar puntos que, a pesar de estar bajo transformaciones afines, puedan ser localizados de manera semejante que en SIFT pero con la premisa de agregar velocidad en el cómputo empleado. SURF [BTG06] evita la representación espacio-escala de tipo piramidal utilizando el concepto de imágenes integrales propuesto en [VJ01], las cuales son la suma de todos los pixeles precedentes[5] delimitados por cualquiera de los filtros propuestos por Viola y Jones[VJ01] generando una pequeña región que es usada para generar a su vez, la imagen integral 2.20.

$$I_{integral}(x,y) = \sum_{x'=0}^{x}\sum_{y'=0}^{y} I(x^{'},y^{'}). \tag{2.20}$$

De esta manera, no se tiene que hacer un submuestreo de la imagen para lograr la representación, ni es necesario operar entre el resultado de cada iteración para obtener la DoG. Sólo se necesita operar con filtros de convolución representando las derivadas parciales de segundo orden de una gaussiana. Cada octava se alcanza variando el tamaño del filtro que van desde los 9×9, 15×15, 27×27, ..., siendo la escala inicial $\sigma_{surf} = 1.2$, mientras que para 27×27,es de $\sigma_{surf} = 3.6$. Cabe señalar que al ser una representación local, no es necesario realizar el cómputo sobre todos los pixeles de la imagen, sino sólo sobre aquellos que sean vecinos de los determinados por el operador Fast-Hessian. La detección de puntos se lleva a cabo a través de la aproximación del determinante de la matriz Hessiana mostrada en la ecuación 2.2:

$$det(\mathcal{H}) \approx D_{xx}D_{yy} - (0.6D_{xy})^2 \tag{2.21}$$

[5]Suponiendo como abstracción de la representación digital de una un imagen uma matriz de mxn y que el origen sea la esquina superior izquierda elemento $(0,0)$. Los pixeles precedentes a un punto (x, y) son todos aquellos que están a la izquierda de x y en la parte superior de y; formalmente: $\{a,b|a<x, b<y, a>0, b>0\}$

donde 0.6 es un resultado aproximado obtenido al calcular la norma de Frobenius normalizada para toda escala.

En este caso, D_{xx}, D_{yy}, D_{xy} son aproximaciones de DoG. Dado que la representación espacio-escala no ha sido submuestreada, no es necesario realizar la localización del punto: ésta es directa. Puesto que la escala está determinada por múltiplos de $\sigma_{surf} = 1.2$, la localización para σ_{surf} tiene que ser determinada por el sistema propuesto por Brown y Lowe mostrado en la ecuación 2.17 también usado en SIFT, aunque sólo se realiza para el parámetro σ_{surf}.

2.2. Descriptores de puntos

2.2.1. SIFT: Transfromada de características invariantes a escala

Para constuir el descriptor SIFT es necesario encontrar un conjunto de orientaciones y de todas ellas, la mas importante a partir de la siguiente expresión :

$$\theta(x,y) = \arctan\left(\frac{D(x,y+1) - D(x,y-1)}{D(x+1,y) - D(x-1,y)}\right) \tag{2.22}$$

$$m(x,y) = \sqrt{(D(x+1,y) - D(x-1,y))^2 + (D(x,y+1) - D(x,y-1))^2} \tag{2.23}$$

.

Operando en cada punto importante, el cual ha sido previamente detectado en la representacion espacio-escala $L(x,y,\sigma_L)$, con sus vecinos, se genera un histograma con 36 intervalos que determinan cada uno una rotación de 10°. A cada muestra se le asigna un peso, correspondiente a la magnitud del gradiente m y una ventana gaussiana cuyo parámetro σ será de $1.5\dot{\sigma}_L$. En el histograma generado, se detecta el valor más alto y todos aquellos que estén a 0.8 del mismo. Esta ventana es rotada en la dirección dominante con la finalidad de comparar todos los puntos en la misma posición. Una vez rotada, una ventana gaussiana de $20\sigma_L$, es subdividida en regiones de 4×4 obteniendo en cada una su orientación principal y gradiente, los que serán almacenados en el descriptor cuya dimensión será de $4 \times 4 \times 8 = 128$.

2.2.2. SURF: Características robustas y aceleradas

Al igual que en SIFT, el primer paso consiste en encontrar una orientación dominante, la que está inscrita en una región circular al rededor del punto de interés. Como segundo paso se alínea a una región cuadrada, de la cual se extrae el descriptor de ella.

Este se logra calculando las respuestas tipo Haar-Wavelet de los vecinos en una región de

radio $6\sigma_L$. El tamaño de la ondeleta depende también de la escala, de tal forma que si la escala es grande, la ondeleta también lo es. Estas respuestas son después pesadas por una ventana gaussiana con $\sigma = 2.5\sigma_L$. La orientación dominante se calcula al sumar todas las respuestas en la ventana y en una orientación deslizante con ángulo 60°.

El tamaño de la ventana para extraer el descriptor es de $20\sigma_L$; después es subdividida en regiones de 4×4 y en cada una de ellas se calculan las respuestas en puntos espaciados 5×5, tanto en la dirección horizontal d_x, como en la dirección vertical d_y. Éstas son aculmuladas en cada subregión y conforman los elementos del descriptor:

$$v = (\sum d_x, \sum d_y, \| \sum d_x \|, \| \sum d_y \|). \tag{2.24}$$

De esta forma, tenemos para cada 4×4 elementos existen 4 características, formando un vector de longitud 64. Para optimizar el tiempo de cómputo, se identifica el *signo* del Laplaciano[6]. Únicamente se almacena un -1.0 o un 1.0; de no ser iguales en el momento de la búsqueda de correspondencia, no se realizan los cálculos posteriores.

[6] traza de la matriz Hessiana.

Capítulo 3

Esquema general del sistema de visión

3.1. Localización

Como mecanismo para introducir el problema de localización espacial de un objeto podemos enunciar un momento general en la vida cotidiana: encontrar algún objeto familiar al momento de despertar. Para poder llevar a cabo esta tarea, necesitamos conocer un estimado, del lugar en el que aquél se puede encontrar. Supongamos que al borde de la cama; lo que hacemos es ubicarnos en el borde de la misma, mirar hacia el piso y comenzar a barrerlo con la vista hasta encontrarlo. Generalmente hacemos ésto, si somos diestros, de izquierda a derecha y de ser zurdos, en sentido inverso. Durante este proceso, casi siempre mantenemos nuestra atención en el centro de nuesta visión; incluso si encontramos un cúmulo de objetos, que en ese momento no tienen importancia, primero centramos nuestra vista en él y hacemos una búsqueda — casi inconsciente — por regiones del mismo, de la manera en la que correspondiente a lo que nuestro cerebro esté acostumnrado.

El porqué del procedimiento mencionado está en el sistema de visión humana [RP99]: en el centro óptico de nuestro ojo, sobre la superficie en la que se proyecta nuestro entorno, llamada fóvea, se encuentra la mayor cantidad de conos — células encargadas de la visión *a color* — las cuales colaboran igualmente en el proceso de definición o nitidez de un objeto. La mayor concentración de bastones — células encargadas de la visón *en blanco y negro* — se encuentra, en cambio, lejos del centro óptico, a pesar de que éstos son más sensibles a la luz, sin embargo, los conos, son quienes tienen mayor resolución temporal. Otro ejemplo cotidiano que puede ser usado para verificar este dato, es tratar de realizar una lectura adecuada cuando existen condiciones de iluminación deficientes.

Tomando este mismo esquema, el sistema de visión de este trabajo, recupera información mediante la localización de *regiones*, en las que se realizará la búsqueda de correspondencias. Los dos enfoques, denominados estáticos y dinámicos, aquí mencionados fueron incluidos como parte de los comportamientos utilizados por el sistema VirBot [JSC99], para el uso de un robot móvil de servicio en la competencia Robocup [TvdZ07]. Para comenzar, la búsqueda del objeto en un entorno desconocido, es realizada a través de regiones dinámicas la cual se revisará en la sección 3.1.1, maximizando la probabilidad de encontrar el objeto a partir de los puntos importantes. Una vez que el objeto ha sido localizado, la forma más sencilla y práctica de llegar a él es a través de movimientos del tipo "movimento pequeño a la izquierda" o "avanza el paso mínimo" , los cuales son transmitidos de manera interna en el sistema VirBot [JSC99] al árbitro planificador de acciones quien se encarga de determinar los comportamientos correspondientes.

3.1.1. Regiones dinámicas

Estas regiones son el resultado de aplicar el procesamiento que en esta sección se detalla, a la imágen en cuestión, con el objetivo de obtener información sobre el lugar en que visualmente se podría ubicar el objeto, tal como se muestra en la figura 3.1. La región donde se encuentra la mayor cantidad de elementos a buscar es la parte izquierda, región que posee no sólo gran contenido cromático, sino tambien variaciones en el mismo, a diferencia de la parte derecha, la cual puede ofrecer cuando mucho sólo un par de puntos, generados únicamente por algunas esquinas visibles.

De nuevo, al intentar localizar la mejor calidad de puntos descritos para buscar su correspondencia, se utliza el concepto de visión humana: dónde se encuentre la mayor cantidad de puntos que probablemente pueden ser comparados. En este trabajo, estas regiones son delimitadas de la siguiente manera. Primero, se realiza un suavizado pequeño (filtro mediana de 5×5), después se ecualiza el histograma y esta imagen resultante se hace pasar el detector de puntos mencionado en [ST94]. La figura 3.2 muestra tres regiones encontradas en una escena; éstas al ser analizadas podrían contener pixeles que pueden ser interpretados como un objeto de interés.

3.1.2. Regiones estáticas

Este tipo de regiones está determinado por un límite constante que es previamente incorporado al sistema. La estructura más sencilla, resultaría de dividir la imagen en subimágenes de tamaño idéntico: si la imágen original tiene una resolución de 640 x 480 pixeles, deberá ser dividida en 4 imágenes de 320 x 240 subimágenes. El problema de este

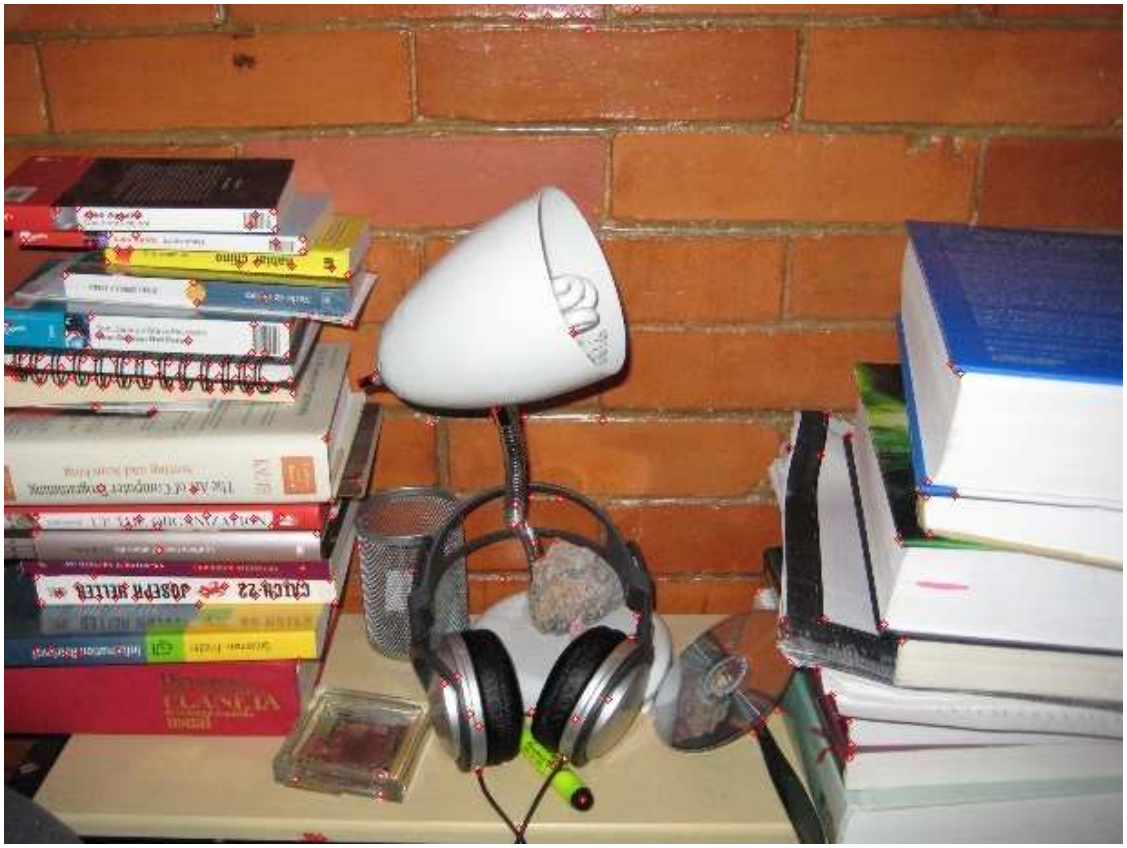


Figura 3.1: Comparación visual: La parte izquierda tiene una mayor cantidad de información visual, por lo que al aplicar alguna función detectora de puntos importantes la cantidad de éstos será mucho mayor (113 puntos) en esta región que la parte derecha de la imagen (26 puntos).

enfoque es que salta a la vista casi inmediatament: el cómputo se intensifica, a pesar de que las imágenes son más pequeñas, este procedimiento se aplicaría 4 veces.

Otro enfoque, un poco más interesante, consiste en generar 5 regiones, una que esté en el centro óptico y que probablemente tenga una superficie mayor que las demás, y 4 regiones yuxtapuestas, en las que se realice el procesamiento no en una secuencia inmediata de cuadros si no únicamente cada cierto numero cuadros . Este enfoque no sólo está dirigido a localizar en una región específica de la imagen. Si no que sigue inspirándose en el sistema de visión humana tal como es descrito en [SWP05].

3.2. Marcado

El marcado es el proceso por medio del cual, una vez identificado el objeto, si se quiere realizar seguimiento, se hace necesario ubicar la región espacial en la imagen analizada. Esto es importante no solo para el seguimiento correspondiente, si no para minimizar también la carga de cómputo que se presenta constantemente ya que al encontrar una región espacial que corresponde a un objeto, la cantidad de elementos de búsqueda sobre las regiones que no han sido marcadas para el seguimiento, tendrán un conjunto de vectores patrón que no tendrán que ser comparados.

Como se ha mencionado, el marcado es el encargado de "actualizar" lo que corresponde al seguimiento; En un sistema de visión se inicializan y actualizan constantemente variables internas como lo es el parámetro σ, parámetro esencial en la búsqueda del espacio-escala, y la posición relativa del objeto en la escena visual, la cual podría utilizarse para realizar Simultaneous Localisation and Matching (SLAM) en otro tipo de aplicaciones. Esta σ puede referirse como a la representacion compuesta en SIFT [Low04] o bien a la que es usada en SURF [BTG06].

La figura 3.2 también ilustra este comportamiento. Una vez que la región ha sido encontrada, se realiza una estimación de la siguiente posición ya sea marcando el centro de la región o usando alguna de las aristas de la región encontrada, junto con sus propiedades, tales como lo son el ancho y el largo. Estas propiedades estarán determinadas dinámicamente con parámetros relacionados al nivel de escala en la que fueron ecntradas.

3.3. Búsqueda de correspondencia

3.3.1. Correspondencia Estándar

La búsqueda de correspondencias o apareamientos, se realiza a través de calcular la distancia cuadrada Euclidiana[1] descrita en la ecuación 3.1, utilizando el conjunto de descrpitores de puntos intenresantes que han sido localizados en una imagen con el conjunto de descriptores que previamente han sido almacenados siendo el patrón de referencia.

$$\begin{aligned} \hat{x} &= (x_1, \ldots, x_n) \\ \hat{y} &= (y_1, \ldots, y_n) \\ d^2 &= (x_1 - y_1)^2 + \cdots + (x_n - y_n)^2 \end{aligned} \tag{3.1}$$

[1] En el caso de este sistema de cómputo, no tiene sentido calcular la raíz cuadrada.

donde $\hat{x}$ es el vector que representa el descriptor de un punto interesante y $\hat{y}$ un vector que representa el descriptor de un punto que ha sido generado en la etapa de entrenamiento.

El umbral de correspondencia que hace posible determinar si los vectores comparados son o no *el mismo* es definido como la diferencia entre la distancia más corta y una constante k que escala a la segunda menor distancia. ($minima\_distancia < k * segunda\_minima\_distancia$). Esta estrategia de búsqueda es presentada en SIFT [Low04]; la constante k se recomienda que no sea mayor a 0.8 y en SURF [BTG06] adquiere un valor empírico de 0.7 . Estos valores numéricos son resultado de las evaluaciones presentadas tanto en [Low04], como en [BTG06] o [MS05], donde la k toma el máximo valor permisible si se considera que las funciones densidad de probabilidad que representan las correspondencias clasificadas como correctas y las clasificadas como incorrectas, tienen un punto de intersección en el que valores en el dominio mayores a éste punto, representarían tener mayor cantidad de resultados con falsa correspondencia.

3.3.2. Arboles K-d

Como se mencionó anteriormente en la sección 3.3.1 la búsqueda de correspondencia está definida a partir del concepto métrico de distancia Euclidiana, tratando de encontrar aquélla que sea *"más cercana"* al vector en cuestión. Si se considera un espacio métrico de dos dimensiones, entonces se trataría de un par de radios de acción que intentarían encontrar a sus vecinos más cercanos. Siendo que los descriptores de características locales están representados por un vector n-dimensional, elegir una mejor herramienta de búsqueda resulta de rigurosa necesidad.

Los árboles K-d y su variante RndK-d [DCM98], son una generalización a los árboles de búsqueda binarios convencionales, pero ofrecen resultados computacionalmente más eficientes para operaciones de búsqueda tal como "*el vecino más cercano*"; conocidas como *peticiones asociativas*[2], las cuales son respondidas en $O(\log n)$.

En consecuencia, si los vectores a tratar están en general definidos[3] por $n = 64$ sólo serían necesarias 5 operaciones ($log(n) - 1$) para determinar el vecino más cercano. Un par de implementaciones que vale la pena mencionar es la realizada en el desarrollo del sistema BaZar [LLF05], el cual es un sistema de realidad aumentada utilzando otro descriptor de características locales llamadas *ferns* las cuales están basadas en un clasificador bayesiano así como en la implementación de SIFT realizada por Rob Hess [Hes].

[2] associative queries.

[3] En [Low04], como en [BTG06] no existe gran diferencia entre descriptores de mayor dimensión.

3.3.3. Cuantización vectorial

La cuantización vectorial es la generalización de la cuantización escalar, utilizada principalmente en la conversión de señales analógicas a formato digital, como ejemplo de una sola variable. Siendo extendida al caso n-dimensional, lleva a técnicas que han resultado en un gran éxito para distintas ramas del ámbito eléctrico o electrónico; tal es el caso de la compresión para la transmisión de senãles [Pro06], o bien la navegaión de un robot móvil a través de la generación de mapas de Voronoi [BM88] los que generalmente son llamados cuantizadores del *vecino más cercano.*

Un Cuantizador vectorial (CV), para el caso de tuplas de mas de un elemento, constituye una herramienta clave para el cómputo de los datos descritos en cada vector: reducción del espacio dado. Tal como en el caso escalar, un rango de valores puede ser determinado por un único valor llamado símbolo, ayudando a la codificación o compresión de datos; extendiendo este concepto, un conjunto de valores, como es el tipo *vecino más cercano* puede ser reducido a un conjunto de vectores de m-dimension dado que $m < n$. Formalmente, en [GG92] se enuncia el siguiente teorema:

Teorema 1. *Para cualquier sistema de codificación que lleva un vector de señal a una de las N palabras binarias y reconstruye el vector aproximado de ésta palabra binaria, existe un CV, con un diccionario de código*[4] *del mismo tamaño que proporciona exactamente el mismo rendimiento de reconstrucción.*

Un CV Q de dimensión k y tamaño N es una relaciión del vector en un espacio Euclideano k-dimensional $\Re^k$ a un conjunto C que contiene N salidas llamadas palabras de diccionario. definido de la siguiente manera:

$$Q : \Re^k \rightarrow C$$

donde $C = (y_1, y_2, \ldots, y_N)$ y $y_i \in \Re^k$ para todo $i \in J = 1, 2, \ldots, N$. Asociado a cada punto en N existe una partición o celda:

$$R_i = x \in \Re^k | Q(x) = y_i$$

la cual cumple la siguientes condiciones $\forall i \neq j$:

$$\bigcup R_i = \Re^k$$

$$\bigcap R_j = 0.$$

[4] Codebook

Ya que los descriptores, tanto SIFT [Low04] como SURF [BTG06], tienen una representación n-dimensional, vale la pena realizar una comparación de rendimiento[5] utilizando cuantización vectorial. En el capitulo 5 se muestra esta comparación para el caso del algortimo SURF usado en este trabajo. La regla de codificación descrita en [BM88] y mostrada en la tabla 3.1 que minimiza la menor distorsión por mínimos cuadrados es:

$$d(x, y) = ||x - y||^2 \tag{3.2}$$

se define un cuantizador de característica del *vecino más cercano* con la siguiente partición con $x \in \Re^k, y \in C, \forall i, j \in \{0, \cdots, N\}$ donde para cada valor de i

$$\forall j, i \neq j | d(x, y_i) \leq d(x, y_j) \rightarrow Q(x) \in R_i$$

Tabla 3.1: Algoritmo Simple para Voronoi

1. $d = d_0$, $j = 1$, $i = 1$
2. Calcular $D_j = d(x, y_j)$
3. Si $D_j < d$, $D_j \rightarrow d$. $j \rightarrow i$
4. Si $j < N$ $j + 1 \rightarrow j$ regresa a 2.
5. Termina. $index = i$

3.4. Seguimiento

3.4.1. El Filtro de Kalman

El filtro de Kalman [Kal60] es una solución al problema conocido como filtrado de mínimos cuadrados, específicamente a través de minimizar el error cuadrático medio Minimum Mean Square Error (MMSE). Norbert Wienner, en los inicios de la decada de 1940, propuso una solución que únicamente consideraba una variable — en ese momento, una sola señal — y las siguientes suposiciones [RG97]:

1. Tanto la señal como el ruido que se sume a ella, tienen características con antelación conocidas, por ejemeplo: Los procesos ruidosos son siempre de carácter Gaussiano.

2. La mejor solución al filtrado de mínimos cuadráticos es siempre óptimo.

[5] Asumiendo independencia entre cada elemento del vector.

Para 1960[6], Rudolf Emil Kalman, basado en la teoría de control, propuso la solución generalizada para el problema del filtrado por mínimos cuadrados, a través de un procedimiento iterativo, el cual considera un modelo de estados que determina la naturaleza del fenómeno tratado y las características estadísticas del proceso tanto en su medición como en su comportamiento, tomando en cuenta que únicamente se almacena el último estado. El conjunto de ecuaciones que define esta solución son presentadas en la ecuación 3.3 como el predictor y en 3.4 como vector de medida,

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{3.3}$$

$$z_{k+1} = Hx_k + v_k \tag{3.4}$$

donde A es la matriz de transición entre el estado actual x_k y el estado siguiente x_{k+1}, B la matriz de control donde u_K es un vector de control, v y w dos vectores que representan el ruido en el proceso y la medición respectivamente; ambos se consideran independientes $(E[w_k, \mathrm{v}_k^T] = 0)$, con distribución normal y de carácter blanco. H es la matriz que relaciona el estado actual con su proceso de medición.

Dado que la naturaleza del filtro actúa con características estadísticas, su estudio está relacionado con el conjunto de probabilidades asociadas al fenómeno en el que se consideran errores antes de realizar la predicción y después de ésta, llamados errores *a priori* y *a posteriori*, agregando a éstos también son considerados los errores durante el proceso de captura o reestructuración de los datos. Dicho de otra forma, existen dos errores que el filtro tendrá en la estimación, uno que será determinado por la probabilidad conocida y otro que estará en función de los valores capturados en el estado actual:

$$\begin{aligned} e_k{}^- &= x_k - \hat{\mathrm{x}}_k^- \\ e_k &= x_k - \hat{x}_k \end{aligned} \tag{3.5}$$

donde con el superíndice $^-$ se denota que este error pertenece al la probabilidad *a priori*, mientras que la $\hat{x}$ denota pertenecer al estimado de la posición siguiente con su probabilidad asociada mostradas en las ecuaciones 3.6

$$P_k{}^- = E[e_k^-, e_k^{-T}] \tag{3.6}$$

$$P_k = E[e_k, e_k{}^T] \tag{3.7}$$

donde P_k denota la probabilidad en el estado k, el superíndice $^-$ se ha comentado con aterioridad, $E[x, y]$ es el operador esperanza matemática, donde x y y son variables aleatorias de probabilidad conjunta.

[6]20 años después de que Wienner propusiera su solución.

Realizando la suposición de que tanto el vector de ruido y el vector de medida en las ecuaciones 3.3 y 3.4 respectivamente, son constantes, se genera la siguiente relación que determina el verdadero problema de minimización:

$$\hat{x}_k = \hat{\mathrm{x}}_k^- + K_k(z_k - H\hat{\mathrm{x}}_k^-) \tag{3.8}$$

en la que K es conocida como la ganancia o el factor de estiramiento[7], y el término $(z_{k+1} - H\hat{x_k}^-)$ es conocido como el residuo o la inovación de medida. Este conjunto de ecuaciones son planteadas en [RG97] como el resultado de formalizar la probabilidad *a posteriori* del estado estimado $\hat{x}_{k+1}$ como una combinación lineal de los estimados *a priori* $\hat{\mathrm{x}}_k^-$ Obteniendo como conclusión la siguiente relación:

$$K_k = \frac{\mathrm{P}_k^- H^T}{H\mathrm{P}_k^- H^T + R} \tag{3.9}$$

de donde fácilmente se puede intuir que si

$$\begin{aligned} \lim_{R_{k\to 0}} K_k &= H^{-1} \\ \lim_{\mathrm{P}_k^- \to 0} K_k &= 0 \end{aligned} \tag{3.10}$$

Al realizar este procedimiento para el caso computable (discreto) tenemos dos conjutos de ecuaciones que dividen al filtro en dos componentes: actualización por tiempo y corrección por mesura[8]:

Tabla 3.2: Ecuaciones por actualización de tiempo

$$\left| \begin{array}{c} \hat{\mathrm{x}}_k^- = A_k + Bu_k \\ \mathrm{P}_k^- = AP_kA^T + Q_k \end{array} \right|$$

Tabla 3.3: Ecuaciones de actualización para la medición

$$\left| \begin{array}{c} K_k = \mathrm{P}_k^- H^T(H\mathrm{P}_k^- H^T + R)^{-1} \\ \hat{x_k} = \hat{\mathrm{x}}_k^- + K_k(z_k - H\hat{\mathrm{x}}_k^-) \\ P_k = (I - K_kH)\mathrm{P}_k^- \end{array} \right|$$

Se utilizan como inicio las ecuaciones por actualización de tiempo, de donde los valores correspondientes son usados en las ecuaciones de mesura y de nuevo se realiza la predicción

[7] *blending factor.*

[8] *time update, meseaurement update.*

como inicio del proceso iterativo. Q y R denotan la covariancia del ruido $p(w)$ y ruido de medición $p(v)$, las que al ser considerados constantes y de carácter gaussiano, tienen una respuesta normal.

El uso de este filtro en el sistema de visión consiste en ubicar la posición del centro de la región marcada como objeto encontrado en el siguiente frame, agilizando la búsqueda, ya que en ésta la búsqueda consiste en ubicar sólo los puntos correspondientes al objeto previamente encontrado y no en todos los descriptores que corresponden a otros objetos. Búsqueda que seguirá en el área restante.

3.5. Sistema de visión: Bloques

El sistema en este trabajo está dividido en dos compomentes: entrenamiento y reconocimiento. La primer etapa, espera una imagen o un conjunto de imágenes en formato digital conocido, ya sea de manera individual, en un directorio o bien tomando una secuencia de cuadros en la captura en tiempo real, en figura 3.3 se detalla com mayor detenimiento éste consta en preprocesar el área seleccionada (imagen o región de ella) que consta en suavizar la imágen por medio del escalamiento y reducción en pirámide gaussiana para luego realizar ecualización del histograma. La segunda fase del sistema, es el reconocimiento, fase en la que se obtiene la imagen de entrada realizando el mismo preprocesamiento que en el entrenamiento mencionado en la sección 3.1.1. Posteriormente, se verifica si se ha encontrado previamente objetos, de ser afirmativo, ellos y su región circundante serán excluidos en la nueva búsqueda, en la que se estima la siguiente posición a través del filtro de Kalman mencionado en la sección 3.4.1. El último procedimineto es generar una máscara de búsqueda que contempla la siguiente posición de los objetos encontrados, si es que éstos existen, agilizando de esta manera el procedimiento de búsqueda en las reguiones fuera de la máscara. La figura 3.4 muestra el proceso.

Figura 3.2: Localización dinámica: En este ejemplo tres regiones han sido detectadas utilizando el enfoque dinámico; en la primer imagen, de la superior a la inferior, se muestra una pequeña región que coincide estar sobre un objeto; en la segunda imagen siguiendo el mismo orden, es localizado una región la cual no coincide con ningun objeto, pero es detectado por la variacion de intensidades que se encuentran sobre la proyección captada por la cámara de la región correspondiente; en la tercer imagen, otra región coincide con un objeto.

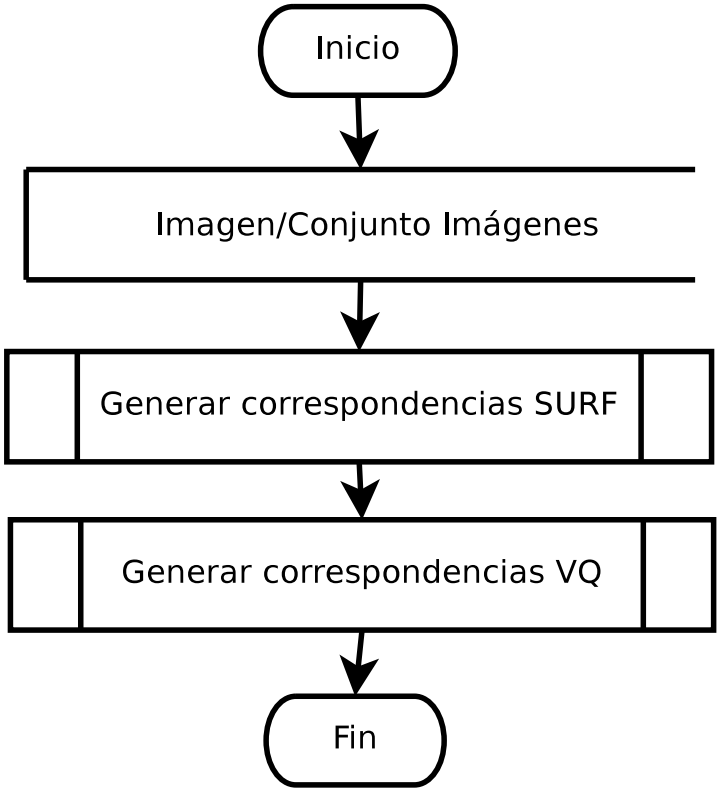


Figura 3.3: Esquema general de la fase de entrenamiento: El primer paso consiste determinar si el entrenamiento estará basado en un conjunto de imágenes o si sólo será una sola imagen. Una vez determinado, se generan todas los vectores SURF posibles, en el caso de ser un conjunto de imágenes en cada una de ellas.

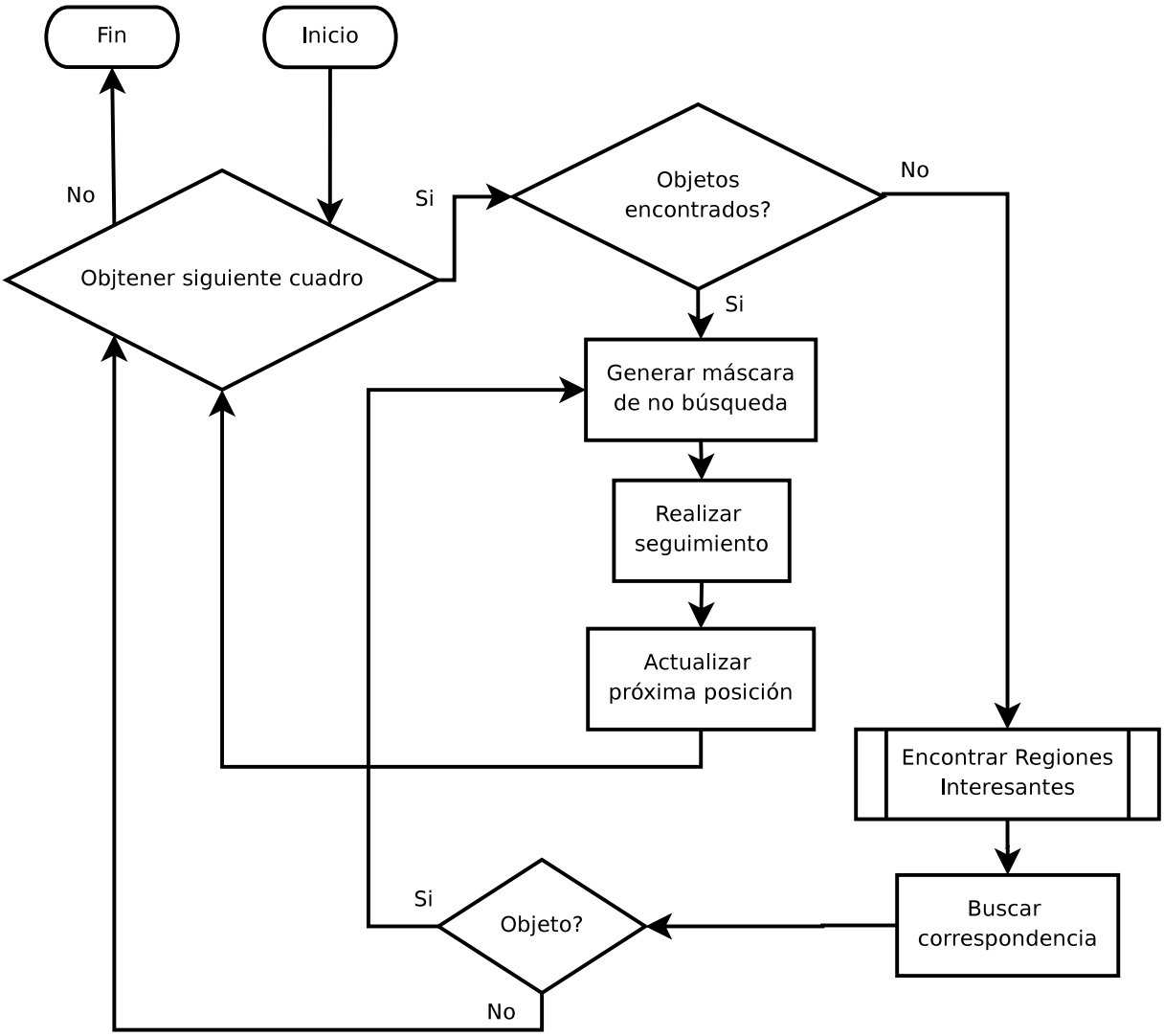


Figura 3.4: Esquema general del sistema en la fase de reconocimiento

Una vez que exista un cuadro que pueda ser procesado, el sistema se pregunta si anteriormente ha encontrado algún objeto, de ser positiva la respuesta, realiza una máscara de no búsqueda, usada para minimizar el computo de las regiones donde no se ha encontrado objeto, ya que de no tener ningún objeto encontrado en menoria, se realiza la búsqueda sobre toda la imagen, buscando correspondencias sobre todos los vectores patrón. sin embargo al tener un objeto en cierta región encontrado, se realiza el seguimiento, actualizando la nueva región de búsqueda y comparando únicamente con los vectores patrón que corresponden al objeto previamente encontrado.

Capítulo 4

Plataforma técnica

En casi todas las referencias, se hace a un lado la implementación técnica de cualquier sistema de cómputo, aludiendo en todo momento al formalismo que involucra el desarrollo del mismo. En ocasiones, inherente a este formalismo se propone o modifican algoritmos que demuestran su eficacia a través de herramientas como la notación asintótica, descrita por Paul Bachmann desde 1884 [Knu97], refiriéndose estrictamente a un resultado que puede ser fácilmente comparado sin considerar limitantes en el proceso de implementación de los mismos. Es cierto que no todo resultado, incluso numérico, tiene que estar sujeto a las reglas específicas de un equipo de cómputo [CLR03]; sin embargo, también es cierto que una mala implementación puede acarrear pésimos resultados [Kle05].

Otra razón por la que considero importante hacer un paréntesis en el escalón técnico es que la ingeniería en cómputo está siendo cada vez más demeritada o absorbida por otras áreas, como la ingeniería de software, ya sea por el perfil del campo laboral mexicano [CON07], o bien por el efecto social [Sta04] en el que estamos inmersos. El abuso en el uso de algunas herramientas de muy alto nivel, lo mismo que ciertos lenguajes de programación con una gran cantidad de bibliotecas *precocidas* han logrado que la calidad del profesionista se vea no sólo disminuida sino totalmente dependiente de las mismas [1]. Esto tiene graves consecuencias tanto en el ámbito académico como en el económico, como lo fué el fracaso llevado por el proyecto Enciclomedia[2].

4.1. Capa física

La capa física o *hardware*, es la herramienta directa con la que un profesional de tecnologías de la información Tecnologías de Información (TI) trabajará toda la vida.

[1] en muchos casos completamente privativas

[2] http://es.wikipedia.org/wiki/Enciclomedia

Coloquialmente, la capacidad de los procesos industriales que permiten la gran integración de grandes sistemas eletrcónicos en menores superfices, conocida Ley de Moore [Moo65], hace pensar que no tiene sentido hablar sobre las carecterísticas actuales del hardware A pesar de esto, creo conveniente, no sólo como un retrato de la situación actual, sino como una meta al mejorar los resultados posteriores considerando que los resultados actuales estan acotados por el poder de cómputo y herramientas aquí descritas, con la finalidad de que en trabajos por venir, se tenga una justa comparación y muestra del verdadero avance alcanzado.

Los corporativos que monopolizan el mercado electrónico definen la facilidad de uso y los alcances de cada algoritmo propuesto. Ejemplo de esto, es el hardware especializado para el PDI que haía sido usado principalmente en sistemas de comunicación y procesamiento de audio, sin embargo la masificación de dispositivos móviles de menor tamaño asi como las tendencias en utilizar aplicaciones conjuntas en audio, video y ambientes virtuales tanto en la forma en la que se transmiten, almacenan y sobre todo la demanda de que todas éstas se ejecuten en tiempo real ha generado que estos corporativos ubiquen un nuevo mercado proporcionando nuevas herramientas tal es el caso de los equipos de alto rendimiento en computo (HPC). Donde la implementación de algunos algoritmos no sólo se vuelve mas sencilla, si no que el tiempo de procesamiento se ve disminuido notablemente.

4.1.1. Equipo de cómputo

El sistema que se propone en esta tesis hace uso de una extensión proporcionada por Intel Corp. [Int05] del tipo Single Instruction Multiple Data (SIMD), conocida como Streaming Single Instruction Multiple Data Extensions (SSE), una evolución de las antiguas MultiMedia eXtensions (MMX); este conjunto de instrucciones fue implementado, debido a que gran parte de los algoritmos actuales realizan un intenso uso de operaciones tipo multiplica y acumula (MAC) sobre regiones perfectamente definidas en memoria. Un par de ejemplos pueden ser que las implementaciones [Pro06] para un filtro digital, o bien la transformada de Fourier discreta; ambas con uso directo en aplicaciones propias del procesamiento digital de señales e imágenes como lo son las aplicaciones de diseño asistido por computadora (CAD), codificadores y decodificadores de audio y video entre otras.

En el sistema propuesto, son usadas las extensiones SSE3[3], incorporadas al procesador Pentium IV, utilizando registros de 128 bits para operaciones de flotante sencillo, por lo que cuatro operaciones pueden ser realizadas en el intervalo de ejecución de una sola instrucción.

El tipo de procesador en el que se ejecutó el sistema es uno de doble núcleo, tipo

[3]disponibles en cualquier procesador del tipo x86 y x86_64.

`Genuine Intel(R) CPU T2050 @ 1.60GHz`[4] . Memoria virtual disponible: 902704 KB.

4.1.2. Adquisición de video

La adqusición de video con la que se trabajó en este sistema se divide en dos tipos. El primero es fuera de línea es decir, se puede obtener la entrada por medio de un archivo que esté soportado por la biblioteca Xine[dta]. El segundo, por captura en tempo real a través de una cámara que pueda ser soportada por Video for linux (v4l) [dtc] o el conjunto de bibliotecas soportadas por el bus ieee 1394: libdc1394 o libraw1394.

Durante la etapa de desarrollo se utilizó una cámara modelo `Unibrain Fire-i Digital Camera`, la cual hace uso de un sensor CCD del fabricante Sony con un modelo ICX098BQ [SC01].

4.2. Capa lógica

La capa lógica o *software* es el conjunto de abstracciones y algoritmos que se ejecutan sobre la capa física, cuya finalidad esta en procesar cierto volúmen de datos obteniendo a partir de este proceso interpretaciones especifícas al resultado del mismo.

Actualmente, no puede existir una concepción de *hardware* sin *software*, incluso ahora cuando una gran cantidad de nuevos dispositivos son actualiazados a través del llamado microcódigo: código que permite modificar el compartamiento del Hardware a través de alguna interfaz de software [PIS84].

Uno de los términos muy socorridos en el área de la ingeniería de software es el de *"Crisis del Software"*, que podria enunciar algo como la Ley inversa de Moore, pero en Sofware. Edsger W. Dijkstra en su artículo, *"The humble programmer"* [Dij72], hace mención a la misma, lo interesante es que fué escrito en el año de 1972. Ahora, las cosas no difieren tanto, y es por esto que vale la pena revisar que estamos *corriendo* en nuestros equipos.

4.2.1. Sistema Operativo

El sistema operativo (SO), es el centro de un sistema de cómputo. Es el que gestiona los elementos de entrada y salida de todas las aplicaciones asi como ser el encargado de garantizar — de estar bien escrita — que la aplicación se ejecute con la prioridad deseada. Al utilizar herramientas de glsSL, el sistema de visión hace uso también del SO Gentoo [Fou] , el cual es una distribución de carácter GNU is not Unix (GNU) con kernel tipo Linux.

[4]Resultado de ejecutar la instrcción CPUID.

En este caso se decidió utilizar un kernel sin parches de tiempo real, ya que el soporte de Hardware necesario para la aplicación, no lo permitía; sin embargo, queda a consideración de trabajos futuros probar su desempeño con un kernel con esta característica, o bien, intentar con otro tipo de SO como lo es FreeBSD [MQ96] (microKernel). El tipo de manejador de memoria usado en lugar del tradicional fue SLUB [htt07] dado que según lo propuesto, incrementa de 5 % a 10 % la eficiencia del intercambio de memoria.

4.2.2. Bibliotecas

Según Tannenbaum [TW06], un SO tiene que ser programable. Es decir, las interacciones hombre/máquina tienen que estar definidas al interior de un conjunto estándar de llamadas a sistema que permita configurar y programar el sistema en la medida que se necesite. Considerando también, que muchas de las operaciones habituales, como puede ser accesar al contenido de un archivo, subir[5] cierta cantidad de información a memoria o abrir algún canal de comunicación, son tan repetitivas que tienen que estar almacenadas sin modificaciones constantes, para ser llamadas en cualquier momento. A este conjunto de funciones y la forma en la que son almacenadas se les conceptualiza como Bibliotecas.

4.2.2.1. OpenCV

Durante los últimos años de la década de los 90's, Intel Corp. propuso una biblioteca de manipulación genérica de imágnes a la que llamó Intel Processing Library (IPL).

En esos años existían otro tipo de bibliotecas como ImageMagick [dtb] que intentaban compilar un gran conjunto de formatos y manipularlos todos como el mismo tipo de dato. Sin embargo, el enfoque de este tipo de bibliotecas, consiste en generar una forma de manipular las imágenes como su representación digital sin facilitar opreaciones de carácter científico o académico puesto que estaban presentes otro tipo de bibliotecas tal como lo es la *GNU Scientific Library Gnu Scientific Library (GSL)* [GDT+03].

De esta manera, un grupo de académicos y el centro de investigación de Intel decidió tomar la IPL y extenderla, logrando no sólo unificar todo en un solo tipo de dato, sino también realizar una abstracción sobre otras bibliotecas , como son de captura para procesamiento de video o bien con funciones que permiten realizar PDI de manera bastate sencilla y accesible, además de implementar gran cantidad de algoritmos bastantes exitosos en la literatura como parte nativa de la biblioteca.

[5] Abusando del concepto generado por estra palabra.

Al ser software libre (SL) OpenCV [Cor01], proporciona una manera muy activa y dinámica de atender dudas fundamentales sobre su implementación más allá del código fuente, también a través de su lista de correo en la que para el año 2008, aproximadamente, obtendría cerca de 50 correos diarios.

Cabe mencionar que esta biblioteca posiblemente estará en casi cualquier instalación de SO de casi todos los investigarores en el área en algunos años por venir; muestra de esto son las peticiones del *Google summer of code* en las que se pide por algunos parches adicionales. Uno de ellos consistió en implementar los descriptores de caractersticas invariantes y locales estudiados en este trabajo. Sin mencionar que al finalizar el mismo, ya estaban incluidos en su nueva versión.

4.2.2.2. Intel Performance Primitives

Al ser propuesta incialmente por Intel Corp, OpenCV puede ser compilada con otra biblioteca de la misma compañía que mejora el desempeño, una vez que ésta sea generada sobre el silicio que la misma corporación vende. Como cualquier compañía, está interesada en vender su tecnología, por lo que proporciona de manera gratuita — mas no abierta — esta biblioteca, que aprovecha todas las ventajas del microprocesador, en caso de ser producto del mismo corporativo, a la que nombró como Intel Performance Primitives (IPP).

Cabe senãlar que la licencia restringe el uso de dicha biblioteca en su carácter gratuito, sólo en los momentos en los que el usuario no recibe remuneración econonómica por su uso, por lo que las pruebas que se realizaron con ella, estuvieron fuera del periodo de patrocinio otorgado por Consejo Nacional de Ciencia y Tecnología (CONACyT).

4.2.2.3. Bibliotecas internas

Dentro del Laboratorio de Biorobótica, se ha desarrollado programas destinado al funcionamiento y operación de robots móviles. Estos programas no son de caracter público siendo que aún no se distribuyen. Para la realización de este trabajo, se inició con el proceso para generar una biblioteca que poco a poco podrá ser distribuida de manera libre para uso genérico en el ámbito de robots móviles.

Capítulo 5

Pruebas y Resultados

El objetivo práctico de este trabajo está en proveer a un robot de servicio con capacidades de visión necesarias, como lo es el reconocimiento de objetos o patrones en una escena, para poder realizar las pruebas propuestas por el comite regulador de la liga *@home* en la competencia Robocup [Lea08].

Es importante mencionar que a diferencia del trabajo pasado [Car07], en éste no se busca verificar la invarianza a rotación y escala prevista por un único entrenamiento. Si no la capacidad de reconocimiento en una secuencia aleatoria, es decir, una secuencia que tiene todo tipo de distorsiones no determinísticas: rotación, cambio de escala, inclinaciones y deformaciones de la imagen, ya sea como parte del movimiento del robot o como parte del procesamiento de normalización de la imagen.

5.1. Pruebas Robocup 2008

La primer etapa a superar consiste en dos pruebas llamadas *"fetch & carry"* o "recoje y entrega" y *"lost & found"* o "busca y encuentra". Opcionalmente podrá realizar la prueba "follow me" o "sigueme". El mecanismo para llevar a cabo la prueba es el siguiente: el jurado, durante el día inmediato anterior a realizar la primer prueba elije de algún lugar público (un supermercado, el complejo donde se realiza la competencia, objetos personales de los asistentes, etc..) cinco objetos que serán tomados como base para las pruebas. Dado que el comite conoce la complejidad del problema, se permite entrenar a los robots con éstos objetos de manera libre de tal forma que en el momento de la competencia, el robot sepa que es lo que tiene que encontrar.
En las dos primeras pruebas mencionadas, un objeto de los cinco elegidos se coloca de manera aleatoria en alguna posición de la arena. El robot tendrá que explorar el escenario hasta encontrar el objeto en cuestión emitiendo una clara señal de cuál es el objeto

que ha encontrado. También deberá cumplir con los objetivos específicos de cada prueba especificados en el conjunto de reglas propuestas [cfhl].

Cabe mencionar que todas las pruebas, conceptualmente para el modulo de visión del sistema VirBot [JSC99], tienen el mismo objetivo: determinar la posición espacial de un objeto que ha sido previamente aprendido por el modulo de visión dada cualquiera que sean las condiciones de captura. Para esto, el equipo participante en la prueba tiene tiempo suficiente para configurar los parámetros dinámicos de sus sistemas.

5.1.1. Recoje y entrega

La prueba *"Fetch & Carry"* tiene como fin, encontrar el objeto y hacer uso de algún mecanismo mecánico que le permita recoger el objeto para llevarlo a cierta ubicación propia dentro de lo que es llamado "arena" o el área donde el robot tiene que explorar. Para esta última fase, otro subsistema del Robot tendrá que poder guiarlo en el recorrido. Sin embargo, durante la búsqueda, el Robot tendrá que basar sus decisiones en el sistema de visión.
En esta prueba, el enfoque de regiones estáticas mencionado en la sección 3.1.2 es el más adecuado, ya que en principio, el robot no debe cambiar el ángulo de búsqueda. Una vez que el objeto es encontrado, el sistema de visión tiene que guiar al robot para colocarse en la posición que mecánicamente pueda usar para agarrar el objeto.

5.1.2. Busca y encuentra

Esta prueba, es un poco más dinámica que *"Fetch & Carry"*, ya que en ésta lo único que se tiene que realizar es dar aviso que algún objeto ha sido encontrado. No es necesario una parte mecánica interactúe con el sistema de visión.
El enfoque de regiones dinámicas es más adecuado, ya que de ser posible si en cierta escala (distancia) el objeto es encontrado, ésta podrá ser usada para guiar al robot, no solo en dirección (ángulo de rotación) sino en distancia.

5.1.3. Sigueme

El robot tiene que atender al llamado de seguimiento de un humano, el que tendrá que trazar cierta ruta qde extremo a extremo por el escenario y recorrela, compitiendo con algún contrincante que en cierto momento tendrá que cruzarse entre el Robot y el humano que esté en ese momento siguiendo; Es en este momento donde el Robot tiene que presentar capacidad robusta de respuesta a oclusiones temporales y poder seguir localizando al humano adecuado.

Esto puede lograrse, entrenando con la vestimenta del humano al robot. Utilizando varias distancias como si fuera distintos objetos. Para tomar ventaja de situaciones en las que el humano tenga que caminar *"más allá"* del rango de reconocimiento limitado por una sola escala.

5.2. Interpretación de resultados

Los resultados que en este trabajo se muestran es el tiempo de procesamiento necesario para analizar cada cuadro adquirido dentro de un conjunto de cuadros que se denomina como secuencia aleatoria. Esta secuencia puede ser parte de la captura en vivo del robot, o bien una secuencia almacenada, ésta última forma de adquición facilita el desarrollo del sistema de visión. Dado que durante la ejecución de la prueba el sistema de visión no es el único proceso que se ejecuta en la misma plataforma y sistema operativo, es suceptible a cambios en tiempos de ejecución por lo que los resultados reportados son el promedio arítmetico del tiempo de procesamiento de cada cuadro adquirido. Este promedio es el resultado de iterar cinco veces la prueba en cuestión.

El rendimiento del sistema estará en función de los siguientes parámetros:

1. Número de características locales proporcionadas por el entrenamiento(ver tabla 5.1 mas adelante).
2. Número de caractetrísticas locales comparadas (ver tabla 5.2.1).

Se utilizan dos secuencias aleatorias: En cada una aparecen cuatro objetos piloto. En algunos casos, los objetos aparecen de manera simultánea. Las pruebas se corren en dos distintos tamaños: 480×360 y 320×240, normal y reducido respectivamente.

Para cada secuencia aleatoria en cada tamaño se agrega un elemento de comparación para cada prueba, es decir, el sistema busca el apareamiento tanto por distancia Euclideana como por cuantización vectorial, de las características almacenadas o características patrón, con las características en ese momento encontradas, siendo que en cada prueba son agregados mayor número de características patrón que representan otro objeto. Cabe señalar que en la búsqueda del objeto, si en una regón ha sido localizado cualquiera de los objetos, en esta regón se deja de buscar por los restantes.

Por lo tanto se tienen dos secuencias aleatorias en dos tamaños distintos con la busqueda de correspondencia para dos diferentes técnicas con cinco pruebas en las que se agrega

incrementalmente un cojunto de características patrón, es decir, se realizan 40 pruebas. las que son denominadas como :

- cuatroObjetoTest01480x360
- cuatroObjetoTest02480x360
- cuatroObjetoTest03320x240
- cuatroObjetoTest04320x240
- dosObjetoTest01480x360
- dosObjetoTest02480x360
- dosObjetoTest03320x240
- dosObjetoTest04320x240
- tresObjetoTest01480x360
- tresObjetoTest02480x360
- tresObjetoTest03320x240
- tresObjetoTest04320x240
- unObjetoMuchasPropiedadesTest01480x360
- unObjetoMuchasPropiedadesTest02480x360
- unObjetoMuchasPropiedadesTest03320x240
- unObjetoMuchasPropiedadesTest04320x240
- unObjetoPocasPropiedadesTest01480x360
- unObjetoPocasPropiedadesTest02480x360
- unObjetoPocasPropiedadesTest03320x240
- unObjetoPocasPropiedadesTest04320x240

siendo que el nombre es un tanto intuitivo, no sobra comentar que el postfijo Test01480x360 corresponde a lo que en este documento me refiero como *primer secuencia tamaño normal* mientras que Test02480x360 es *segunda secuencia tamaño normal* siendo que Test03 y Test04 son las pruebas en *tamaño reducido*.

Objeto 1	376
Objeto 2	105
Objeto 3	91
Objeto 4	160

Tabla 5.1: Número de características por objeto

Prueba	PS normal	PS reducido	SS normal	SS reducido
Un objeto	15502	5599	9066	3461
Dos objetos	14743	5677	7306	3620
Tres objetos	15796	5851	16806	5413
Cuatro objetos	15175	5806	11050	3730

Tabla 5.2: Número de caracetrísticas comparadas .PS: primer secuencia. SS: Segunda secuencia.

5.2.1. Resultados numéricos

Tanto en las tablas como las gráficas el tiempo es mostrado en segundos. En cada gráfica se muestra en el eje ordenado el tiempo de procesamiento en segundos mientas que en el eje de las abcisas solo se muestra el número de cuadro correspondiente en la secuencia. En otras palabras lo que se puede observar en cada gráfica es el tiempo de procesamiento por cada cuadro de la secuencia.

Para las pruebas correspondientes a la cuantización vectorial. Un conjunto de 64 vectores característicos son probados. En el caso de la comparación estándar por mínima distancia Euclideana, el número de características por elemento es el mostrado en la tabla 5.1 . Cabe mencionar que se toma como criterio de localización únicamente haber encontrado siete correspondencias con lo que no se sigue realizando la búsqueda sobre todas la demás en la región dada, si esta regla se cumple.

Tabla 5.3: Primer secuencia Tamaño normal.

Descripción	Distancia Euclideana	Distancia Euclideana + IPP	Cuantización vectorial
Un objeto	219.74	167.9	237.7
Dos objetos	228.73	176.52	267.77
Tres objetos	241.42	184.68	268.26
Cuatro objetos	244.92	186.78	267.54

Tabla 5.4: Primer secuencia Tamaño reducido.

Descripción	Distancia Euclideana	Distancia Euclideana + IPP	Cuantización vectorial
Un objeto	93.34	77.17	104.92
Dos objetos	94.41	77.6	114.28
Tres objetos	95.62	78.1	115.31
Cuatro objetos	97.04	79.04	117.58

Tabla 5.5: Segunda secuencia Tamaño normal.

Descripción	Distancia Euclideana	Distancia Euclideana + IPP	Cuantización vectorial
Un objeto	159.71	118.69	178.19
Dos objetos	190.57	2.1224	228.21
Tres objetos	205.37	127.79	229.28
Cuatro objetos	200.54	127.86	240.82

Tabla 5.6: Segunda secuencia Tamaño reducido.

Descripción	Distancia Euclideana	Distancia Euclideana + IPP	Cuantización vectorial
Un objeto	70.63	60.9986	81.2180
Dos objetos	82.88	64.27	84.41
Tres objetos	84.56	67.66	85.26
Cuatro Objetos	85.18	70.4	90.08

En el caso de las pruebas que involucran aprareamiento por CV se analizan la cantidad de faltos positivos y falsos negativos. Tomando un falso positivo como aquella correspondencia encontrada, que en verdad no está en relación con el objeto, así como un falso negativo un apareamiento nulo cuando deberá de haber realizado el apareamiento, considerando que las correspondencias encontradas en el caso de la distancia Euclideana son correctas y la base de esta comparación. El resultado en porcentaje, para la primer secuencia se muestra en la tabla 5.7 y el resultado para la segunda secuencia se muestra en la tabla 5.8 en ambos tamaños para cada prueba.

Secuencia	Primer secuencia	
Tamaño	normal	reducido

+/-	+	-	+	-
Un objeto	68.639798	8.284024	69.650655	23.404255
Dos objetos	69.772727	12.048193	67.682263	27.272727
Tres objetos	68.686869	15.789474	67.748918	33.333333
Cuatro objetos	68.589744	18.032787	70.346320	25.641026

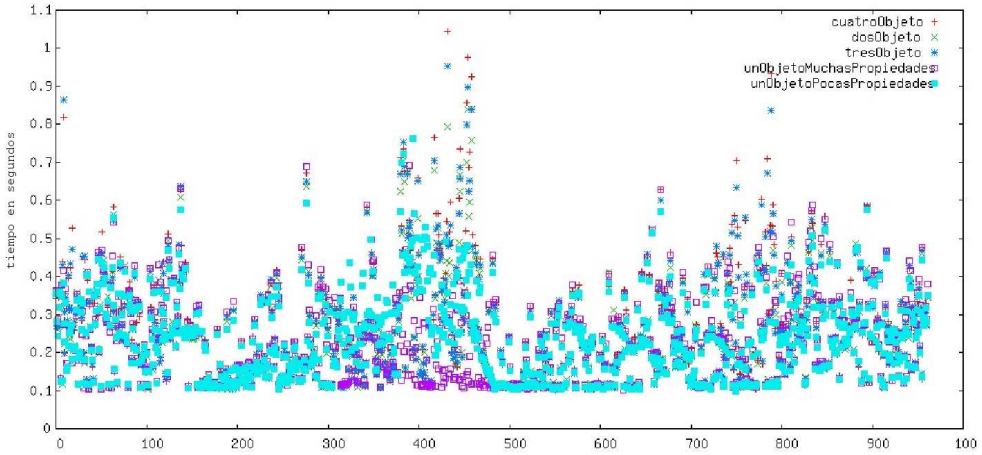
Tabla 5.7: Falsos positivos y negativos para la primer secuencia

Secuencia	Seguna secuencia			
Tamaño	normal		reducido	
+/-	+	-	+	-
Un objeto	74.331551	4.469274	65.600000	4.583333
Dos objetos	71.544715	7.258065	59.175258	15.294118
Tres objetos	71.764706	12.380952	60.980810	15.498155
Cuatro objetos	73.258427	12.881356	58.521561	9.090909

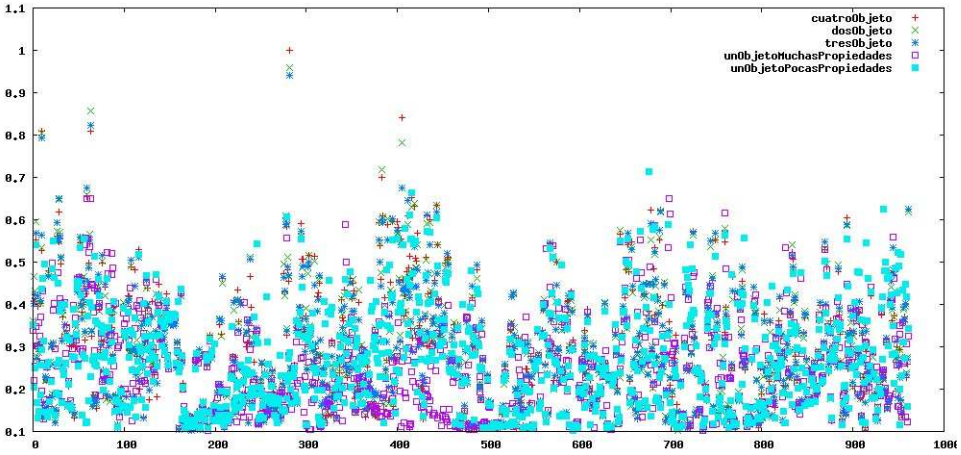
Tabla 5.8: Falsos positivos y negativos para la segunda secuencia

Tanto en la tabla 5.7 como en la tabla 5.8, se puede apreciar que en promedio, por lo menos dos de cada tres resultados encontrados en CV, son falsos positivos, mientras que en presencia del objeto en promedio un 15 % se realiza una falsa detección.

Para cada comparación de resultados, se elabora una prueba t de student considerando que la hipótesis nula en este caso es que la correspondencia por distancia Euclideana tiene menor tiempo de procesamiento que la que corresponde a la correspondencia por cuantización vectorial. El resultado de esta prueba es un numero real en el rango [0,1] llamado estadístico-p. Si el estadístico-p es mayor que 0.05 entonces consideramos que la prueba es verdadera. Lo que significa que la distancia Euclideana tiene mejor desempeño en este sistema que la correspondencia por cuantización vectorial. En caso contrario, valores menores a 0.05 la cuantización vectorial tiene mejor desempeño.

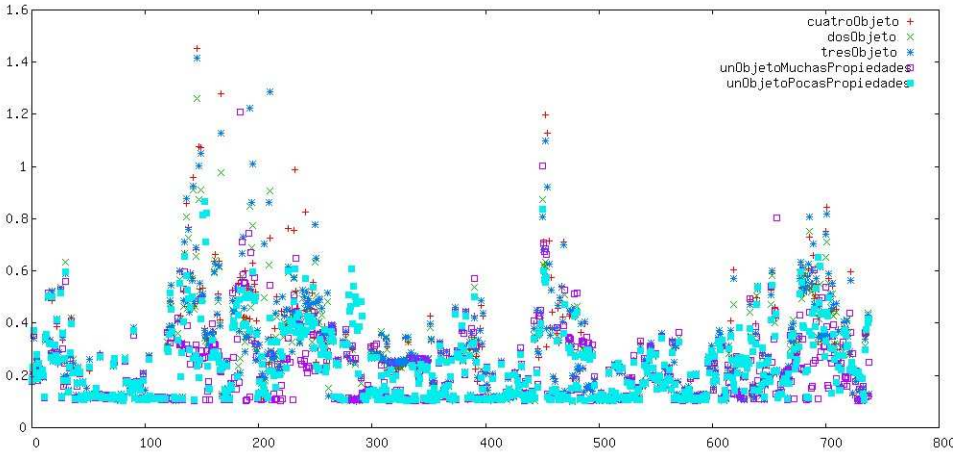


(a) Correspondencia: Distancia Euclideana.

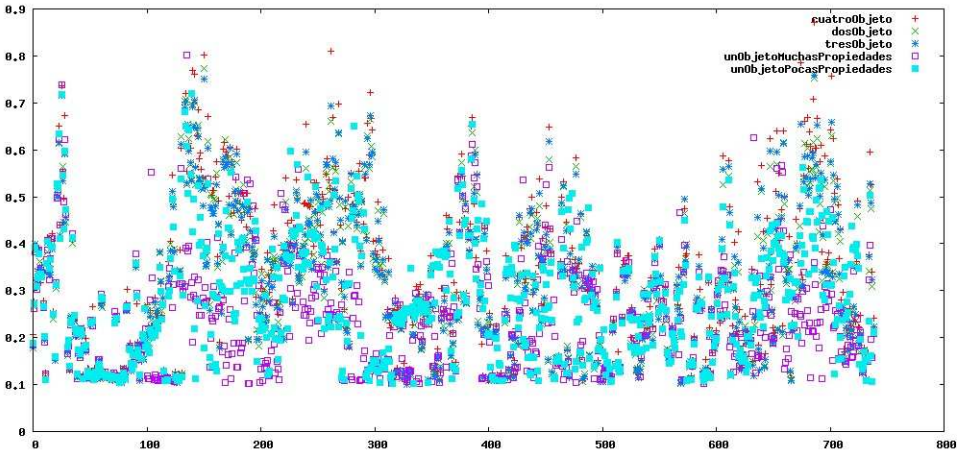


(b) Correspondencia: Cuantización vectorial.

Figura 5.1: En esta prueba, no se percibe gran diferencia entre los dos tipos de comparaciones, sin embargo se alcanza a distinguir que los tiempos son mas uniformes en el caso de la cuantización vectorial. El estadístico-p en este caso es : $\{7.08e^{-5}, 6.119e^{-13}, 1.292e^{-6}, 7.62e^{-5}\}$, por lo que en este caso el desempeño es menor en distancia Euclideana.

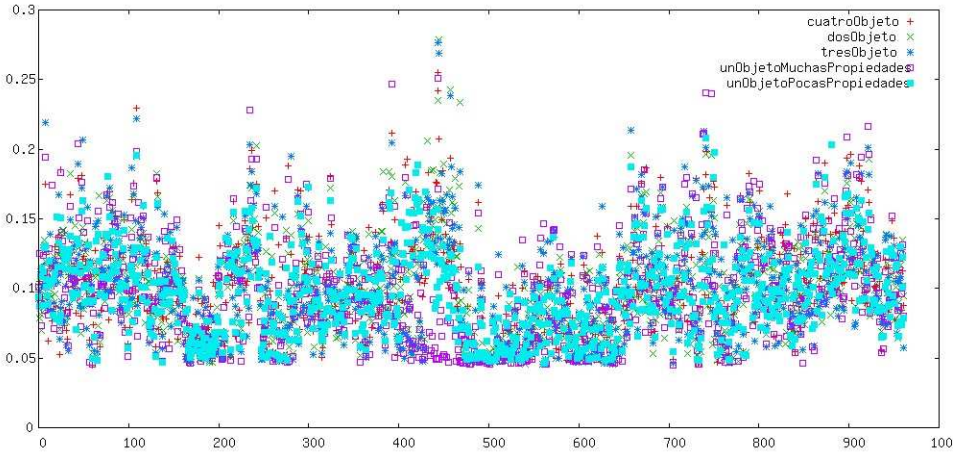


(a) Correspondencia: Distancia Euclideana. Tamaño: 480×360

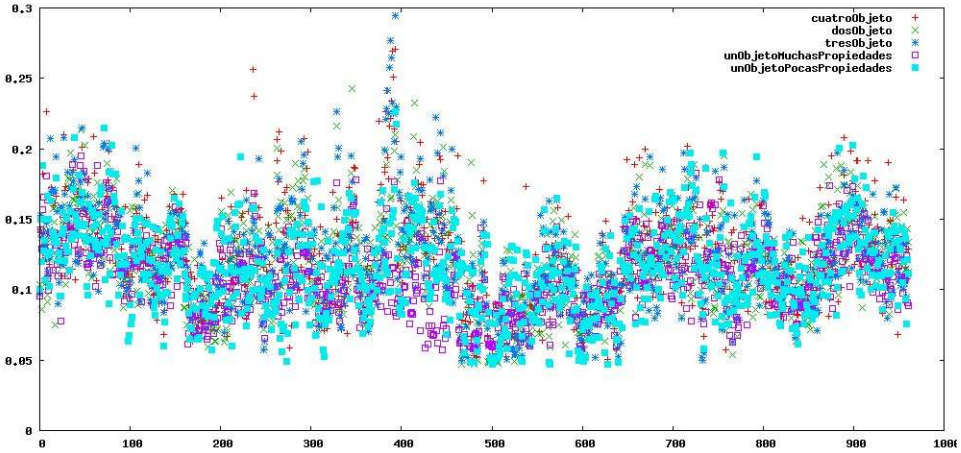


(b) Correspondencia: Cuantización vectorial. Tamaño: 480×360

Figura 5.2: Esta es, la prueba mas distintiva. La tendencia en el caso de cuantización vectorial a mantener de manera uniforme el tiempo de comparación total para la búsqueda de correspondencia se mantiene al existir un límite en el n'umero de comparaciones, los grandes saltos de cuadro a cuadro, en el número de comparaciones no estan presentes. El estadístico-p en este caso es: $\{6.83e^{-10}, 1.99e^{-10}, 1.11e^{-4}, 1.62e^{-14}\}$, por lo que también el desempeño es mejor en distancia Euclideana.

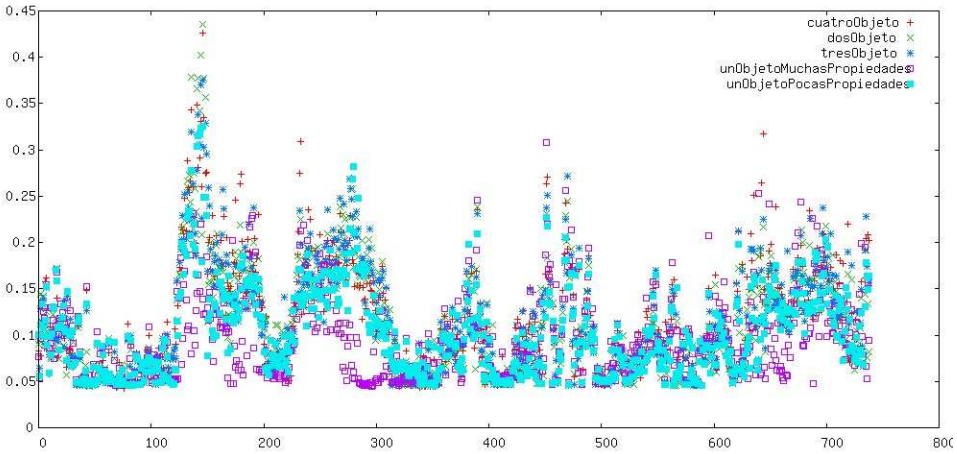


(a) Correspondencia: Distancia Euclideana.

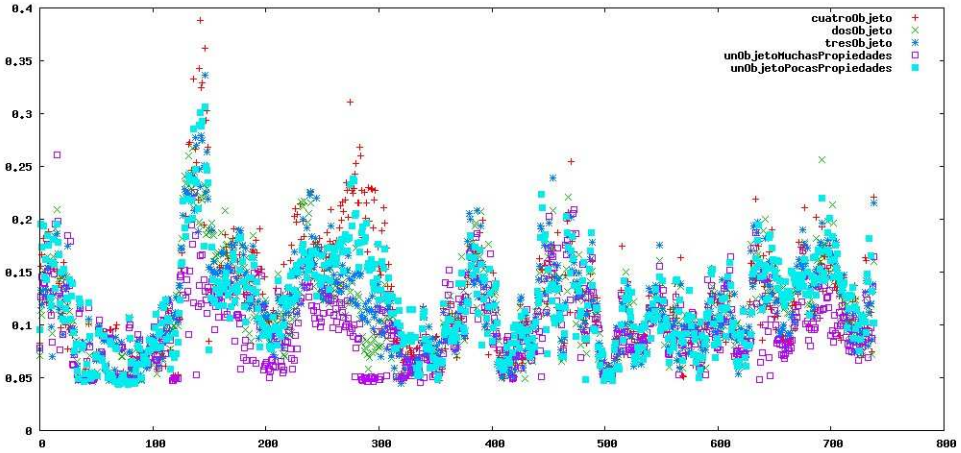


(b) Correspondencia: Cuantización vectorial.

Figura 5.3: Al reducir el numero de propiedades a comparar, puede notarse que no existe gran diferencia entre los dos mecanismos de búsqueda de correspondencia. La tendencia en cuantización vectorial se mantiene. sin embargo el rendimiento no es tan bueno como el caso de la correspondencia por Distancia Euclideana. El estadístico-p en este caso es: {0, $1.66e^{-14}$, $2.18e^{-14}$, $1.62e^{-14}$ }, por lo que el desempeño es mejor en distancia Euclideana.



(a) Correspondencia: Distancia Euclideana.



(b) Correspondencia: Cuantización vectorial.

Figura 5.4: Esta prueba hace notar lo que en pruebas anteriores se ha revisado: se mantiene una distribución uniforme en tiempo de respuesta para la cuantización vectorial, pero sólo es mas eficiente cuando el número de características comparadas es mucho mas grande que para el caso de la Distancia Euclideana. Esto puede notarse en el pico entre el cuadro 100 y 200, donde la cuantización mejora por el límite de comparaciones realizadas. El estadístico-p en este caso es: {0.013, 0.222, 0.365, 0.02 }. A pesar de la comparación visual, en este caso se hace notar que el número de comparaciones es factor para determinar el tipo de estrategia, el mejor desempeño es la cuantización vectorial.

Capítulo 6

Conclusiones

El estado del arte en el campo de visión por computadora respecto al análisis de imágenes o secuencia de ellas a través de características principales[1] que pueden ser locales o globales, en conjunto, permiten relacionarlas con la existencia espacial de un objeto en una escena.

En este trabajo se detalla y hace uso de una técnica para la extracción de características locales llamada SURF [BTG06], que por diseño resulta ser más rápida [TM08] — computacionalmente hablando — y robusta en cuanto a modificación de parámetros que las existentes hasta el momento, tal como lo es SIFT [Low04]. Estos parámetros, al igual que en muchos campos aún no son determinísticos en su totalidad, tal como es la tolerancia al ruido, oclusiones sobre la superfice analizada o bien cambios en la luminancia de la imagen adquirida.

Un factor de riesgo a considerar en el uso de características locales para el reconocimiento de objetos, es la cantidad de vectores[2] que se utilizan y la separabilidad que pueda existir entre ellos en la fase de búsqueda de correspondencias. Estas dos condiciones parecieran obvias para el análisis pero cambios pequeños en el contexto de una escena pueden dar resultado no sólo a falsos positivos, si no a un incremento en la complejidad del sistema que se esté realizado al tratar de discernir entre los elementos que pueden pertenecer o no a la case de un objeto. Tal como el caso de la cuantización vectorial, en la que gran número de vectores patrón incrementan la deficiencia para catalogar un objeto, es decir el conjunto de vectores que generan menores distancias se ve maximizado.

El cambio que se hace más notable, es la intensidad de luz que llega a un objeto, haciendo variar en gran medida los valores que conforman al descriptor. Ésta es una de

[1] No confundir con PCA [Principal Component Analysis]

[2] también llamados características o propiedades, una vez procesado su descriptor.

las perturbaciones más comunes y difíciles de tratar, ya que al normalizar también las condiciones con técnicas comunes del procesamiento de imágenes PDI, éstas hacen variar las condiciones locales de la imágen, generando un descriptor completamente diferente al original. Es decir, no genera correspondencia.

Otro factor de riesgo al utilizar características locales, es que a pesar de ser invariantes a escala y rotación dentro de un determinado límite físico éste puede ser medido en distancia, pero es mas específico hacerlo en relación de aspecto. Cada descriptor puede tener distintos niveles de invarianza, por lo que no es aplicable un determinismo estricto para generar patrones genéricos que contemplen un conjunto de observaciones esperadas.

En lo que se refiere a la búsqueda de correspondencia, dos enfoques se trataron: correspondencia estándar y correspondencia a través de cuantización vectorial. La primera es la más efectiva, ya que a comparación de la segunda, por lo menos si consideramos un conjunto de mas de 7 puntos, también tenemos orientación espacial de la posición del objeto. En cuanto a la segunda: muchos falsos positivos son tomados, por lo que un enfoque probabilístico se tiene que realizar para poder descartarlos. Sin embargo, la cuantización permite un rápido seguimiento, a través de un rango previamente definido. El problema de la cuantización vectorial, está en el número de comparaciones necesarias cuando no se ha localizado un objeto ya que no es una comparación única en un espacio limitado si no es un ejercicio superior, dependiente del numero de vectores característicos a operar, en este trabajo se utilizo un diccionario de 64 vectores característicos, por cada característica encontrada en la imagen.

En vista a los resultados obtenidos en este trabajo, dos elementos importantes a considerar cuando se utilizan características locales son:

1. Cantidad de características definidas en la fase de entrenamiento del sistema: encontrar las características que hacen verdaderamente discernible o representable al objeto en todo el conjunto de características descritas como patrón.

2. Resolución o tamaño de la adquisición: Dado que los detectores, tanto SIFT como SURF buscan propiedades del tipo grandes cúmulos, verificar las condiciones externas[3] que puedan afectar los valores del descriptor dramáticamente.

A pesar de que la complejidad computacional de ambos algoritmos se mantiene constante [TM08] — aún cuando es utilizada la técnica de búsqueda de correspondencia mas elemental: distancia euclideana —, el número de características elegidas para ser comparadas resulta el factor mas trascedental.

[3]como puede ser el entorno en un robot móvil.

6.1. Trabajo Futuro

- Realizar un estudio sobre el tipo de características que son afectadas por el cambio de intensidad sobre una imagen. Este estudio tendrá como objetivo buscar la forma de normalizar la entrada en la región de búsqueda así como la de encontrar una medida sobre la que se pueda elegir automáticamente los puntos más significativos en un vecindario.

- Incorporar parámetros en el descriptor que unan a ésta característica con un conjunto de las mismas. Es decir exhibir a un descriptor como un conjunto de descriptores a pesar de ser analizados individualmente. Un acercamiento utilizando teoía de grafos es adecuado para aplicaciones fuera de línea, sin embargo para aplicaciones de tiempo real, conceptualizar la idea de un megadescriptor suena apropiado.

- Para el caso de cuantización vectorial, separar el descriptor en cada una de sus partes, para ser analizado como un conjunto de propiedades con un significado específico, tal como pueden ser las respuestas en sentido horizontal o vertical únicamente.

- En cuanto al sistema, utilizar los conceptos proporcionados por ambos algoritmos, tanto en su representacion espacio/escala, como en la búsqueda y supresion de máximos locales con la finalidad de parametrizar una búsqueda genralizada con poco procesamiento.

- Incorporar parámetros en el entrenamiento que estén asociados directamente con la aplicación del sistema de visión tal como distancia en un sistema basado en la realidad o la posición absoluta de la camára respecto a una referencia fija.

Apéndice A

Instalación del Sistema

Par instalar el sistema, es necesario tener una instalación correcta de OpenCV. Como se mencionó en el Capťulo 4, OpenCV es una biblioteca que pretende ser portable y no ser especfica del Sistema Operativo. En la dirección electrónica `http://opencv.willowgarage.com/` se podrá encontrar una detallada referencia de como instalarla en cada Sistema Operativo. Este trabajo contempla la version 1.0.0 de la Biblioteca. Futuras versiones de éste sistema podrán descargarse del sitio `http://gitorious.org/projects/localfmcvs`.
Este trabajo se desarrolló bajo una plataforma Linux invitando también al lector a utilizar éste sistema como base de su trabajo y prueba. Por lo que se asume se estará usando el mismo. Cabe mencionar que MacOS al ser un Sistema Operativo tipo unix, también esta cubierto por el mismo. Para saber si se tiene una instalación adecuada de OpenCV basta con correr el siguiente comando

```
$ echo -e '#include <cv.h>\n \
int main ( void  ) { \
return printf("\\n\\n%s\\n\\n", \
CV_VERSION); }' | \
gcc `pkg-config opencv --libs --cflags` -x c - 2>/dev/null \
&& ./a.out
```

si el resultado muestra:

```
1.0.0
```

entonces podremos continuar con la instalación del sistema.

Para obtener el sistema mencionado es necesario descargarlo utilizando la herramienta para control de versiones llamada `git`, la cual tiene como objetivo manejar control de versiones.

```
git clone git://gitorious.org/localfmcvs/mainline.git
```

Lo que bajara el sistema al directorio actual.

El primer paso es instalar la biblioteca de Cuatizacion Vectorial, la que está ubicada bajo el directorio pr. Esta bibliteca neceista correr los siguientes comandos en orden:

```
$ aclocal
```

```
$ autoconf
```

```
$ automake -a
```

```
$ autoheader
```

```
$ ./configure && make && make install
```

Una vez que este instalada la biblioteca, Solo bastará regresar el directorio raíz del proyecto y escribir

```
$ make
```

Dos ejecutables serán creados:

- Rrecog
- Rtrain

`Rtrain`: acepta dos argumentos. El primero será del tipo *camara*, *video* o *imagen*. El segundo argumento, será el sufijo con lo que los archivos de cuantización y características serán creados.

`Rrecog`: acepta dos argumentos: El primero es del tipo *camara* o *archivo*. Mientas que el segundo sera el nombre del archivo en caso de que el primer argumento sea *archivo*, la ruta de la secuencia para anlizar.

Lista de Abreviaturas

CCD	Coupled Charged Device	1, 28
CONACyT	Consejo Nacional de Ciencia y Tecnología	30
CV	Cuantizador vectorial	18, 36, 37
DoG	Diferencia de Gaussianas	8–11
GNU	GNU is not Unix	28, 29
GSL	Gnu Scientific Library	29
IA	Inteligencia Artificial	1
IPL	Intel Processing Library	29
IPP	Intel Performance Primitives	30
LoG	Laplaciano de Gaussianas	8, 9
MAC	multiplica y acumula	27
MMSE	Minimum Mean Square Error	19
MMX	MultiMedia eXtensions	27
PDI	Procesamiento digital de imágenes	1, 4, 27, 29, 43
RP	Reconcimiento de Patrones	1, 4
SIFT	Scale Invariant Feature Transform	2, 8, 10, 11, 16, 17, 19, 42, 43

SIMD	Single Instruction Multiple Data	27
SL	software libre	30
SLAM	Simultaneous Localisation and Matching	16
SO	sistema operativo	28–30
SSE	Streaming Single Instruction Multiple Data Extensions	27
SURF	Speed-Up Robustness Features	2, 10, 16, 17, 19, 42, 43
TI	Tecnologías de Información	26
UKF	Unscented Kalman Filter	3
v4l	Video for linux	28

Bibliografía

[Bio] Laboratorio Biorobótica. Laboratorio biorobótica. facultad de ingeniería. unam.

[BL02] Matthew Brown and David Lowe. Invariant features from interest point groups. In *In British Machine Vision Conference*, pages 656–665, 2002.

[BM88] B. Bavarian and F. Mortezaie. Application of the voronoi diagram technique for mobile robot navigation. *Systems, Man, and Cybernetics, 1988. Proceedings of the 1988 IEEE International Conference on*, 1:469–471, Aug 1988.

[BTG06] Herbert Bay, Tinne Tuytelaars, and Van Gool. Surf: Speeded up robust features. May 2006.

[Car07] Gerardo Carrera. Seguimiento de objetos con el uso de caracteriticas locales invariantes a escala y rotacion. Master's thesis, Univerisdad Nacional Autonoma de Mexico, 2007.

[cfhl] Robocup comittee for @home league. Rulset for @home competition.

[CLR03] Thomas H. Cormen, Charles E. Leiserson, and Clifford Rivest, Ronald L. y Stein. *Introduction to Algorithms*. McGraw-Hill Science / Engineering / Math, 2nd edition, December 2003.

[CON07] CONACyT. Informe general del estado de la ciencia y la tecnologia. Technical report, Consejo Nacional de Ciencia y Tecnologia, 2007.

[Cor01] Intel Corporation. *Open Source Computer Vision Library*, 2001.

[DCM98] Duch, Estivill Castro, and Martinez. Randomized k-dimensional binary search trees. 1998.

[DH72] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972.

[Dij72] Edsger W. Dijkstra. The humble programmer. *Commun. ACM*, 15(10):859–866, October 1972.

[dta] Xine developement team. Xine libraries for xine movie player.

[dtb] ImageMagick development team. A collection of tools and libraries for many image formats.

[dtc] Video For Linux development team. Video for linux.

[Fou] Gentoo Foundation. Gentoo linux distribution. http://www.gentoo.org.

[GDT+03] Mark Galassi, Jim Davies, James Theiler, Brian Gough, Gerard Jungman, Michael Booth, and Fabrice Rossi. *Gnu Scientific Library: Reference Manual*, February 2003.

[GG92] Allen Gersho and Robert M. Gray. *Vector Quantization and Signal Compression.* Springer, 1992.

[Gre06] Gary Bishop GregWelch. An introduction to the kalman filter, July 2006.

[Hes] Robert Hess. Sift implementation. http://web.engr.oregonstate.edu/ hess/.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. pages 147–151, 1988.

[HSSB98] Mike Heath, Sudeep Sarkar, Thomas Sanocki, and Kevin Bowyer. Comparison of edge detectors: A methodology and initial study. In *Initial Study,Çomputer Vision and Image Understanding*, pages 38–54. IEEE Computer Society Press, 1998.

[htt07] http://lwn.net/. The slub allocator. April 2007.

[Int05] Intel. *IA-32 Intel Architecture Software Developers Manual.* Intel, 2005.

[JSC99] Alistair Holden Jesus Savage Carmona, Mark Billinghurst. Virbot: Un sistema para operar robots de servicio. 1999.

[Kal60] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, (82 (Series D)):35–45, 1960.

[Kle05] Eva Kleinberg, Jon Tardos. *Algorithm Design.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[Knu97] Donald E. Knuth. *Art of Computer Programming, Volume 1: Fundamental Algorithms (3rd Edition).* Addison-Wesley Professional, July 1997.

[Lea08] Ropbocup League. Robocup league home page: http://www.robocup.org, 2008.

[Lin91] Tony Lindeberg. *Discrete Scale-Space Theory and the Scale-Space Primal Sketch.* May 1991.

[LLF05] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2:775–781 vol. 2, 2005.

[Low03] D. Lowe. Distinctive image features from scale-invariant keypoints. 2003.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[MF94] J. Leitao M. Figueiredo. Sequential and parallel image restorarion: neural network implementations. 1994.

[Moo65] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8):114–117, April 1965.

[Mor80] Hans Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. (CMU-RI-TR-80-03), September 1980.

[MQ96] Karels McKusick, Bostic and Quarterman. *The Design and Implementation of the 4.4BSD Operating System.* Addison-Wesley Longman, Inc, 1996.

[MS01] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, 1:525–531 vol.1, 2001.

[MS02] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. *Computer Vision ECCV 2002*, pages 128–142, 2002.

[MS05] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005.

[PIS84] C. A. Papachristou, V. R. Immaneni, and D. B. Sarma. An automatic migration scheme based on modular microcode and structured firmware sequencing. *SIGMICRO Newsl.*, 15(4):155–164, 1984.

[Pro06] Dimitris K. Proakis, John G. y Manolakis. *Digital Signal Processing (4th Edition).* Prentice Hall, March 2006.

[RG97] Patrick Hwang Robert Grover. *Introduction to Random Signals and Applied Kalman Filtering.* John Wiley, 3th edition, 1997.

[RP99] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. 1999.

[SC01] Wfine CCD Corporation Sony Corporation. Sony ic098bq data sheet, 2001.

[SH85] Linda G. Shapiro and Robert M. Haralick. A metric for comparing relational descriptions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-7(1):90–94, 1985.

[ST94] Jianbo Shi and C. Tomasi. Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994.

[Sta04] Richard M. Stallman. *Free Sofware for Free Society.* GNU Press, 2004.

[SWP05] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. volume 2, pages 994–1000 vol. 2, 2005.

[TM08] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: A survey. 2008.

[TvdZ07] Paul G. Ploger Tijn van der Zant. Lightweight management - tamning the robocup development process. 2007.

[TW06] Andrew S. Tanenbaum and Albert S. Woodhull. *Operating Systems Design and Implementation (3rd Edition) (Prentice Hall Software Series).* Prentice Hall, January 2006.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 1:I–511–I–518 vol.1, 2001.